Vorwort

Der vorliegende Tagungsband enthält die wissenschaftlichen Papiere zu den im Rahmen des MMBnet2015-Workshops am 10./11. September 2015 an der Universität Hamburg präsentierten Vorträgen. Über die vergangenen Jahre hinweg hat sich die MMBnet-Workshop-Reihe inzwischen sehr gut etabliert und ein erfreulich hohes wissenschaftliches Niveau erreicht. Ausgehend von ersten noch eher unregelmäßig stattfindenden Veranstaltungen, die damals noch als sogenannte Arbeitsgespräche realisiert wurden – Veranstaltungen ohne jegliches Programmkomitee und relativ informell – ist man inzwischen bei den nunmehr regelmäßig in 2-jährigem Turnus stattfindenden MMBnet-Workshops angekommen mit einem hochkarätigen Programmkomitee, das bei seiner jetzigen Ausgabe mit Vertretern aus sieben unterschiedlichen Nationen auch durchaus einen deutlichen internationalen Anstrich erhalten hat. MMBnet2015 ist bereits der achte Workshop dieser Reihe und wird wiederum, wie sämtliche seiner Vorgänger-Workshops, am Fachbereich Informatik der Uni Hamburg durch den Arbeitsbereich Telekommunikation und Rechnernetze (TKRN), im Namen der GI/ITG-Interessengruppe "Messung, Modellierung und Bewertung von Rechensystemen (MMB)", organisiert. Leider ist in Anbetracht der bevorstehenden Pensionierung des Initiators und lückenlosen MMBnet-Veranstalters (Prof. Wolfinger) im Herbst 2016 zu befürchten, dass die achte Veranstaltung des MMBnet-Workshops auch gleichzeitig die letzte sein könnte. Aber vielleicht findet sich noch eine Kollegin oder ein Kollege bzw. eine Gruppe von Kollegen, der/die bereit ist bzw. die bereit sind, die Tradition der Hamburger MMBnet-Workshops fortzusetzen, denn die Thematik der Workshops verliert zurzeit keineswegs an Relevanz, wie bereits der CfP des aktuellen MMBnet-Workshops deutlich macht, der Formulierungen beinhaltete wie:

Kommunikations-/Rechnernetze und verteilte Systeme gewinnen weiterhin zunehmend an Bedeutung (z.B. in Form von sozialen Netzen, virtuellen Netzen, 'Cloud Computing'-Systemen, u.v.a.m.) – gleichzeitig nimmt ihre Komplexität jedoch auch erheblich zu. Besonders deutlich wird diese Tendenz bei Hochgeschwindigkeitsnetzen, dem (Future) Internet sowie bei der Mobil- und Echtzeitkommunikation. Zwar ist weiterhin eine enorme Leistungssteigerung bei der Rechnerhardware (Multi-Core), den Vermittlungsrechnern (optische Vermittlung) und vor allem bei der Datenübertragung (sowohl bei optischer als auch bei funkbasierter Übertragung) zu verzeichnen. Andererseits steht diesen Fortschritten jedoch ein sehr starkes Wachstum bei den Benutzeranforderungen gegenüber. So gibt es beispielsweise im Kontext des Internet zurzeit die zum Teil relativ schwierig zu lösenden Probleme: Bereitstellung mobiler Internetzugänge für eine extrem hohe Anzahl von Benutzern über sehr unterschiedliche Zugangsnetze, verstärkter Wunsch nach Multimedia- und Echtzeitkommunikation, immer mehr und häufig komplexe verteilte Anwendungen bzw. anwendungsorientierte Dienste (z.B. Video on Demand, Streaming, VoIP) sowie evtl. gravierende Engpässe in den Anschlussnetzen und/oder den Endsystemen. Daher wird der Bedarf nach Bewertung von Kommunikationsnetzen und verteilten Systemen keineswegs überflüssig werden, sondern es werden in verstärktem Maße neue Mess-/Modellierungsmethoden und -werkzeuge erforderlich werden, u.a. zur Berücksichtigung der Dienstgüte (QoS bzw. QoE), des Performance-Managements und steigender Anforderungen an die Zuverlässigkeit und Verfügbarkeit.

Ziel des Workshops war es deshalb, zum einen *Weiterentwicklungen der Grundlagen und Werkzeuge zur Messung und Modellierung* aufzuzeigen und zum andern über *Erfahrungen bei interessanten Anwendungsstudien* (zu Rechnernetzen oder verteilten Anwendungen) zu berichten. Gewünscht waren im CfP auch innovative Beiträge, die eine Brücke schlagen zwischen den bislang noch häufig getrennten "Welten" der Leistungs- und der Zuverlässigkeitsbewertung. Erwünscht war seitens der Veranstalter überdies eine besonders starke Beteiligung am Workshop von Nachwuchswissenschaftlern (insbesondere Promovierende und jüngere Postdoktoranden) mit FuE-Aktivitäten im MMB-Umfeld.

Die für den MMBnet2015-Workshop angestrebten Themenschwerpunkte betrafen bei den

- *Methodischen Grundlagen und Werkzeugen:*
  - formale Beschreibungstechniken (zur Spezifikation von System-, Last - und Zuverlässigkeitsmodellen),
  - Last-/Verkehrsmodellierung und -charakterisierung, Lasttransformation, 'Traffic Engineering',
  - mathematisch-analytische, simulative, hybride/heterogene Modellauswertung,
  - Modellierungswerkzeuge für Netzanalysen, Netzemulatoren, Last-/Verkehrsgeneratoren,
  - Messmethoden und -werkzeuge;

sowie bei den

- *Anwendungsbereichen (bezogen auf die Bewertung durch Messung oder Modellierung):*
  - Netzarchitekturen: Software-Defined Networking, Network Function Virtualization, Future Internet, Internet of Things, Ubiquitous Computing, Pervasive Computing, Service-oriented Computing, Grid Computing, Cloud Computing, Content-Distribution-Netze, Peer-to-Peer-Netze, Mobile Services, Quadruple Play, etc.
  - Mobilkommunikation und drahtlose Netze: MANET, VANET, LTE, WLAN, Ad-hoc-Netzwerke, Sensornetze, Body-Area-Netze, RFID-Systeme, etc.
  - Verteilte multimediale Anwendungen: E-Learning, E-Health, Telepresence, Verkehrstelematik, Videokonferenzen und Verteiltes CSCW, Video-Streaming, Video-on-Demand, Soziale Netze, etc.
  - Echtzeitkommunikation: VoIP, Videoconferencing, Online-Gaming, Live IPTV, etc.

Die akzeptierten Beiträge sind auf drei Sitzungen verteilt worden zu den folgenden Themengebieten:

- Messung und Modellierung für softwaredefinierte Netze
- QoE-Bewertung für Anwendungen im Internet und in Mobilnetzen
- Programmier-Framework und analytische Verlässlichkeitsmodelle.

Unser herzlicher Dank gilt den Mitgliedern des Programmkomitees für die zügige und sorgfältige Begutachtung der eingereichten Papiere, so dass für jeden eingereichten Beitrag mindestens drei Begutachtungen (im Mittel sogar knapp vier) vorlagen, die das Auswahlverfahren signifikant unterstützten. Den Vortragenden danken wir für die termingerechte Erstellung der Endversionen ihrer Beiträge (wobei sämtliche Autoren ein spezielles Lob verdienen für die allseits sorgfältige Berücksichtigung der aus dem Review-Prozess resultierenden Anregungen und Modifikationswünsche der Gutachter).

Zur Aufwertung der Veranstaltung wurden dieses Mal sogar insgesamt drei eingeladene Beiträge in das Workshop-Programm aufgenommen, allesamt aus dem akademischen, universitären Umfeld und allesamt geliefert seitens sehr ausgewiesener Persönlichkeiten, insbesondere durch die Professoren Dr. Hans Daduna (Uni Hamburg), Dr. Dr. h.c. mult. Paul J. Kühn (Uni Stuttgart) sowie Dr. Peter Reichl (Uni Wien). Da die wissenschaftliche Ausrichtung dieser eingeladenen Vortragenden auch in wichtige Nachbargebiete der Informatik wie Angewandte Mathematik und Nachrichtentechnik reicht, konnte erreicht werden, dass die eingeladenen Vorträge nicht nur wissenschaftliche Tiefe sondern auch eine signifikante fachliche Breite aufweisen. Für die Bereitschaft der „Invited Speakers“, nach nur minimaler Bedenkzeit unser Angebot anzunehmen, den eingeladenen Vortrag zu halten (gepaart mit einem zusätzlichen Beitrag für den Tagungsband), möchten wir uns an dieser Stelle nochmals ganz herzlich bedanken, zumal es sich, wie bereits erwähnt, bei sämtlichen dieser Kollegen um wirklich hochangesehene Vertreter (und das nicht nur national!) innerhalb der „MMB-Community“ handelt.

Die tatkräftige Unterstützung der Workshop-Vorbereitung und -Durchführung seitens Frau Katrin Köster (TKRN-Sekretariat) verdient überdies äußerst lobende Erwähnung. Nicht zuletzt möchten wir auch unserem eigenen Fachbereich und speziell unseren Kolleginnen aus der Bibliothek herzlich Dank sagen für die Unterstützung des Workshops insbesondere durch die Publikation des Tagungsbandes als Fachbereichsbericht (sowohl in gebundener Papierform als auch in elektronischer Form). Nur durch diese großzügige Unterstützung war es wie bereits bei sämtlichen Vorgänger-Workshops der MMBnet-Reihe wieder möglich, auf die Erhebung einer Teilnahmegebühr völlig verzichten zu können – im Sinne einer (auch finanziellen) Unterstützung unseres wissenschaftlichen Nachwuchses im MMB-Umfeld, der durch die MMBnet-Workshops gezielt gefördert werden soll.

Als Organisatoren des MMBnet2015-Workshops wünschen wir uns, dass unser Plan, durch eine relativ strenge Limitierung der Teilnehmeranzahl (insbesondere Begrenzung auf die Autoren und Koautoren der akzeptierten Beiträge, die TPC-Mitglieder sowie weitere Personen mit nachgewiesener signifikanter Erfahrung im MMB-Umfeld) sowie durch die Gewährung von viel Freiraum für intensive Diskussionen erneut dazu führen wird, dass während des Workshops ein Klima entstehen wird, das zu einem intensiven Wissensaustausch zwischen sämtlichen Teilnehmern sowie zu deren fachlicher Vernetzung führen wird. Mögen besonders auch die beim Workshop anwesenden Doktorandinnen und Doktoranden für ihre Dissertationsprojekte wertvolle Anregungen und wissenschaftliche Stimuli erhalten und möge der achte MMBnet-Workshop nicht der letzte seiner Reihe sein!

Hamburg, im September 2015 Prof. Dr. Bernd E. Wolfinger, PD Dr. Klaus D. Heidtmann

Mitglieder des Programmkomitees:

- **Abdollahpouri, Alireza** (Univ. of Kurdistan, Sanandaj, Iran)
- **Braun, Torsten** (Univ. Bern, Schweiz)
- **Buchholz, Peter** (Univ. Dortmund)
- **Carle, Georg** (TU München)
- **Daduna, Hans** (Univ. Hamburg)
- **De Meer, Hermann** (Univ. Passau)
- **Fidler, Markus** (Leibniz Univ. Hannover)
- **German, Reinhard** (Univ. Erlangen)
- **Haßlinger, Gerhard** (Deutsche Telekom Netzproduktion)
- **Heckmüller, Stephan** (SSI Schäfer Noell, Bremen)
- **Heidtmann, Klaus-D.** (Univ. Hamburg)
- **Krieger, Udo R.** (Univ. Bamberg)
- **Kühn, Paul J.** (Univ. Stuttgart)
- **Lai, Junyu** (Univ. of Electronic Science and Technology of China, Chengdu, VR China)
- **Lehmann, Axel** (Univ. der Bundeswehr München)
- **Misra, Sudip** (Indian Institute of Technology, Kharagpur, Indien)
- **Reichl, Peter** (Univ. of Vienna, Wien, Österreich)
- **Riedl, Johannes** (SIEMENS AG)
- **Schmidt, Thomas C.** (HAW Hamburg)
- **Schmitt, Jens B.** (Univ. Kaiserslautern)
- **Siegle, Markus** (Univ. der Bundeswehr München)
- **Spaniol, Otto** (RWTH Aachen)
- **Szczerbicka, Helena** (Leibniz Univ. Hannover)
- **Timm-Giel, Andreas** (TU Hamburg-Harburg)
- **Tutschku, Kurt** (Blekinge Institute of Technology, Schweden)
- **Uhl, Tadeus** (FH Flensburg)
- **von Bodisco, Alexander** (Hochschule Augsburg)
- **Waldhorst, Oliver** (Daimler AG, Karlsruhe)
- **Walke, Bernhard** (RWTH Aachen)
- **Wolfinger, Bernd E.** (Univ. Hamburg) ***(Vorsitz)***
- **Zitterbart, Martina** (KIT Karlsruhe)

Zusätzliche Reviewer:

- **Bozakov, Zdravko** (Leibniz Univ. Hannover)
- **Runge, Torsten** (Univ. Hamburg)

Inhalt

Eingeladene Beiträge

- *H. Daduna (Universität Hamburg):*
  Networks of Queues in a Random Environment: A Survey of Product Form Results 1
- *P. J. Kühn (Universität Stuttgart):*
  Novel Methodologies for Real-time Performance Evaluation of NFV-Based Network Infrastuctures 18
- *P. Reichl (Universität Wien):*
  Back to the Future: Towards an Internet of People (IoP) 23

Messung und Modellierung für softwaredefinierte Netze

- *D. Raumer, F. Wohlfart, D. Scholz, P. Emmerich, G. Carle (TU München):*
  Performance Exploration of Software-based Packet-Processing Systems 27
- *T.M. Runge, F. Wohlfart, D. Raumer, B.E. Wolfinger, G. Carle (Univ. Hamburg / TU München):*
  How Do Multiple Network Cards Influence the Software Router Performance? 37
- *M. Hock, M. Willems (Karlsruher Institut für Technologie):*
  Koexistenz nicht-kompatibler TCP-Varianten in softwaredefinierten Netzen 46

QoE-Bewertung für Anwendungen im Internet und in Mobilnetzen

- *M. Li, P.N. Tran, A. Timm-Giel (TU Hamburg-Harburg):*
  Radio Resource Allocation in LTE Femtocell Considering QoE 55
- *S. Momeni, S. Weichler, B.E. Wolfinger (Universität Hamburg):*
  Pre-reservation of TV Channels to Improve the Availability of IPTV Services Offered in Vehicular Networks 63
- *H. Jürgensen (Nextragen GmbH), T. Uhl (FH Flensburg):*
  Auswertung von QoS-Messdaten in einer Cloud-Umgebung 75

Programmier-Framework und analytische Verlässlichkeitsmodelle

- *M. Triebe, D. Charousset, R. Hiesgen, T.C. Schmidt (HAW Hamburg)*
  Das C++ Actor Framework im Leistungsvergleich 83
- *R. Krenzler, H. Daduna (Universität Hamburg)*
  Performability Analysis of an Unreliable M/M/1-type Queue 90

Liste der Beitragenden 97

Networks of queues in a random environment: Survey of product form results

Hans Daduna
Department of Mathematics, Center of Mathematical Statistics and Stochastic Processes
University of Hamburg, 20146 Hamburg, Germany
Email: daduna@math.uni-hamburg.de

***Abstract*—This report collects research on various queueing networks in a random environment. The describing processes are Markov and the joint queueing-environment process is separable with product form stationary distribution. We consider autonomous environments, which are Markov for their own, and non-autonomous environments, the development of which is influenced by the network.**

I. Introduction

Queueing networks with product form steady state distribution have found many fields of applications, e.g. production systems, telecommunications, computer system modeling. The success of this class of models and its relatives stems from the simple structure of the steady state distribution which provides access to easy performance evaluation procedures. The central property of these systems is that the stationary distribution is of so-called *product form*, which states that in steady state the joint queue length distribution for any fixed time instant is the independent product of the marginal queue length distributions of the nodes. This property is subsumed often under the term "separable model", and loosely speaking this is characterized as "asymptotically and in steady state the queue lengths decouple".
Starting from the work of Jackson [Jac57] many generalizations have been developed. A branch of research which recently has found interest are queueing networks in a random environment with product form steady state distributions.

The first approach to find product form steady state distributions for Jackson networks in a random environment seemingly was the work of Zhu [Zhu94]. For partly different settings Economou [Eco05], [Eco03], Balsamo and Marin [BM13], and Tsitsiashvili, Osipova, Koliev, Baum [TOKB02] continued the investigations. The general procedure in these papers for a network of single exponential servers is as follows (explained in terms of Zhu's notation): The key stochastic ingredients in the classical Jackson network for node i of the network are an external Poisson-λ_i arrival stream, exponential-μ_i service times, and a Markovian routing scheme, which produces an overall arrival rate η_i and a local marginal stationary distribution ξ_i with

$$\xi_i(n) = \left(1 - \frac{\eta_i}{\mu_i}\right)\left(\frac{\eta_i}{\mu_i}\right)^n, \quad n \in \mathbb{N}_0\,. \tag{I.1}$$

In [Zhu94] the fundamental parameters depend on the state of the external environment. If the environment is in state k the parameters are adapted to $\lambda_i(k), \mu_i(k), \eta_i(k)$, and with the additional assumption that the utilizations $\eta_i(k)/\mu_i(k)$ do not depend on k, $\eta_i(k)/\mu_i(k) = \eta_i/\mu_i$, say, the local marginal stationary distribution ξ_i is (I.1) again. Zhu does not explain how this independence of k for the utilizations emerges, i.e. no mechanism is described which holds $\eta_i(k)/\mu_i(k)$ invariant during changes of the environment.
Tsitsiashvili, Osipova, Koliev, Baum [TOKB02] argue for the invariance $\eta_i(k)/\mu_i(k) = \eta_i/\mu_i$ with the existence of natural control mechanisms and rules of technical and biological systems to react on changes of the environment, but no explicit rule is given.

The quest for such rules has on the other side a long history in related fields, especially in the control of communications networks, where the term "rerouting schemes" describes the necessity to react, e.g., to buffer overflow, broken down nodes, (partial) degrading of transmission lines. Examples from the literature are collected and described in connection to product form

steady states in [SD03] under the heading of "skipping", "repeated service–random destination", "stalling". Skipping is called "jump over protocol" by other authors, see [Dij88]. We describe in Section V some recently found schemes for the network control which react to changing environment and are physically meaningful, see Section V-B. These rerouting schemes encompass "skipping" and "repeated service–random destination" as special cases and are integrated into Jackson networks with environment dependent service capacities, see Section V-C. This implies that the results generalize not only those in [Zhu94], [Eco05], [BM13], and [TOKB02], but also those of [SD03], which are not covered by the results in the before mentioned papers.

We distinguish in this review autonomous and non-autonomous environments. An autonomous environment can be described as Markov process of its own, and influences in a one-way direction the development of the queueing network (Section III). Non-autonomous environments are not Markov in isolation, i.e. there are influences in a two-way interaction of the environment on the network, and vice versa: Jumps of the network may enforce the environment to change its state (Section IV and V). In both frameworks we come up with product form stationary distributions which imply that the network processes and the environment process decouple in steady state for a fixed time instant.

Our main topic in case of autonomous environments is the investigation of networks, where the nodes break down randomly due to external forces. Besides of the classical ergodic networks we investigate Jackson networks where parts of the network can not stabilize, but there is a stable subnet. Another issue are nodes in Jackson networks where idling nodes are filled by an additional "infinite supply". We show that separability emerges in any case.

In Section IV we describe various queueing systems and networks connected with an environment or a second specific system which can be considered as an environment for the queues. In Section IV-A we extend the availability theory of Section III-A by incorporating queue length dependence into the transition rates of the environment. Similar two-way interactions occur when the queue, modeling a production process is coupled with an associated inventory. In Section IV-C we describe a birth-death-loss process in a random environment and indicate how this can be used to investigate a referenced node in a wireless sensor network.

In Section VI we describe very recent results: Random walks and other systems from statistical mechanics are embedded into a Jackson network as environment. This Jackson network can be considered as a disordered medium wherein one or several particles move. The particle and the Jackson network interact in a two-sided fashion. So, the environment is non-autonomous.

In Sections VII and VIII we sketch some topics and models which are closely related to those described in the review, but mostly lack the separability property.

Summarizing: The concern in this review is basically two-fold:
General aim: Our focus is on integrated models which incorporate the network's and the environment's development in a unified Markovian system description; The joint network-environment process.
Special emphasis: To find integrated models which provide us with methods that resemble the Jacksonian product form calculus:
Asymptotically and in equilibrium the queues and the environment as components (in state space $\mathbb{N}^J \times K$) decouple. This will lead us to a stationary joint distribution which is the independent product of the 1-dimensional marginals in space.

Notation and conventions:
$2^M = \mathcal{P}(M)$ is the set of all subsets of a set M.
For sets A, B: $A \subseteq B$ means A is subset of or equal to B, and $A \subset B :\Leftrightarrow (A \subseteq B$ and $A \neq B)$.
Kronecker Delta: δ_{xy} is 1 iff $x = y$, and 0 otherwise.
$\overline{J} := \{1, \ldots, J\}$ is the set of nodes, $\overline{J}_0 := \overline{J} \cup \{0\}$.
e_j is the standard j-th base vector in $\mathbb{N}_0^J$ if $1 \leq j \leq J$. and e_0 is the J-dimensional zero vector.
We write $\underline{n} = (n_j : j \in \overline{J}) \in \mathbb{N}_0^J$.
We set $0/0 := 0$, and for $a > 0 :\ a/0 := \infty$.
$||\gamma||_\infty := \max\{\gamma_j : j \in \overline{J}\}$ for $\gamma = (\gamma_j, : j \in \overline{J}) \in \mathbb{R}_1^J$.
For $\gamma = (\gamma_j, :\ j \in \overline{J})$ and $\eta = (\eta_j, :\ j \in \overline{J})$: $\gamma \wedge \eta := (\min(\gamma_j, \eta_j) : j \in \overline{J})$.
For functions $\gamma, r : K \to \mathbb{R}$ on countable set K we denote by $\gamma \bullet r$ their pointwise multiplication.
For vector v indexed by K we denote by $I_{(v)}$ the diagonal matrix indexed by K with v on the diagonal.

II. JACKSON NETWORKS

Jackson networks are the prototype of stochastic networks (networks of queues) which in steady state show a product form structure. The simplicity of this structure underlies the success of this class of models in many areas, e. g. computer and telecommunication networks and flexible manufacturing systems.

A Jackson network [Jac57] consists of nodes $\overline{J} = \{1, \ldots, J\}$, where indistinguishable customers arrive, are served, possibly at several stations, and eventually depart from the network. The nodes are exponential single servers with state dependent service rates and with an infinite waiting room under first–come–first–served (FCFS) regime. If at node j there are $n_j > 0$ customers present, either in service or waiting, then service is provided there at rate $\mu_j(n_j) > 0$; we set $\mu_j(0) := 0$. All customers follow the same rules.

Routing is Markovian, described in terms of an irreducible stochastic routing matrix

$$r = (r(i,j) : i,j = 0, \ldots, J), \tag{II.1}$$

where the artificial "node 0" represents the source and the sink of all customers. Strict inequality may hold for $r(0,0) \geq 0$, which indicates that some of the arriving customers may be rejected. We set $\overline{J}_0 := \overline{J} \cup \{0\}$.

Customer arrive in a Poisson-λ stream, which is split (independently) according to the first row $r(0) := (r(0,i) : i \in \overline{J})$ of r. Then at nodes $j \in \overline{J}$, we observe independent Poisson-λ_j arrival streams with $\lambda_j = \lambda r(0,j)$. A portion $\lambda_0 = \lambda r(0,0)$ of the arriving customers is rejected (lost).
A customer departing from node i immediately proceeds to node j with probability $r(i,j) \geq 0$, and departs from the network with probability $r(i,0)$. The traffic equations for the admitted customers

$$\eta_j = \lambda_j + \sum_{i \in \overline{J}} \eta_i r(i,j), \qquad j \in \overline{J}, \tag{II.2}$$

have a unique solution $\eta = (\eta_j : j \in \overline{J})$. Note, that (II.2) only counts for the admitted customers because $\lambda_1 + \ldots + \lambda_J = \lambda(1 - r(0,0))$. For $r(0,0) = 0$, r is the so-called extended routing matrix of standard Jackson networks, see [DS08][(3.2)].

Let $\mathbf{X} = (X_t : t \geq 0)$ denote the vector process of joint queue lengths in the network. $X_t = (X_1(t), \ldots, X_J(t)) \in \mathbb{N}_0^J$ reads: at time t there are $X_j(t)$ customers present at node j, either in service or waiting. The assumptions put on the system together with the usual independence assumption imply that $\mathbf{X}$ is a Markov process on state space $\mathbb{N}_0^J$ with generator $Q^{\mathbf{X}} = (Q^{\mathbf{X}}(\underline{n}, \underline{m}) : \underline{m}, \underline{n} \in \mathbb{N}_0^J)$ with positive entries for $\underline{n} \in \mathbb{N}_0^J$ and $i, j \in \overline{J}$

$$\begin{aligned} Q^{\mathbf{X}}(\underline{n}, \underline{n} + e_j) &= \lambda_j, \\ Q^{\mathbf{X}}(\underline{n}, \underline{n} - e_i) &= \mu_i(n_i) 1_{(n_i > 0)} \cdot r(i,0), \\ Q^{\mathbf{X}}(\underline{n}, \underline{n} - e_i + e_j) &= \mu_i(n_i) 1_{(n_i > 0)} \cdot r(i,j), \end{aligned}$$

For an ergodic network process $\mathbf{X}$ Jackson's theorem [Jac57] states that the unique steady-state and limiting distribution ξ on $\mathbb{N}_0^J$ is for $\underline{n} = (n_1, \ldots, n_J)$

$$\xi(n_1, \ldots, n_J) = \prod_{j=1}^{J} \left(C(j)^{-1} \prod_{k=1}^{n_j} \frac{\eta_j}{\mu_j(k)} \right), \tag{II.3}$$

with normalizing constants $C(j)$.

Separability. The product form of the steady state (II.3) is often characterized by calling networks **separable** which exhibit this structure. This means in a precise sense that in a **stationary system** for each **fixed time instant** the joint queue length distribution of the network is the independent composition of the marginal single queue length distributions. This separability is the main feature which makes Jackson networks applicable in easy to perform first-order quantitative analysis of complex systems.

III. NETWORKS IN AN AUTONOMOUS MARKOVIAN ENVIRONMENT

We follow the approach presented by Zhu [Zhu94]: Take standard Jackson networks with joint queue length process $\mathbf{X} = (X(t) = (X_j(t) : j \in \overline{J}) : t \geq 0)$ influenced by a finite environment process $\mathbf{Y} = (Y(t) : t \geq 0)$ with state space K.
Main assumption: $\mathbf{Y}$ is Markov and ergodic, and "independent" of $\mathbf{X}$ (autonomous).
Assumption on the dynamics: If $Y(t) = k$ and $X_j(t) = n_j$, then at time t at station j

- external arrivals are Poisson-$\lambda_j(k)$,
- service is provided with intensity $\mu_j(n_j, k)$,
- routing from j to i is with probability $r(j,i)(k)$.
- the traffic equation

$$\eta_j(k) = \lambda_j(k) + \sum_{i \in \overline{J}} \eta_i(k) r(i,j)(k), \ (\forall \ \ j \in \overline{J}),$$

has a unique, **strict positive** solution.

The following theorem of Zhu was reproved by Economou [Eco05], Balsamo and Marin [BM13], and Tsitsiashvili, Osipova, Koliev, Baum [TOKB02] who demonstrated it as examples of their general investigation of Markov processes in a random environment.

Theorem III.1: [Zhu94] Consider a Jackson network with joint queue length process $\mathbf{X} = (X(t) = (X_j(t) : j = 1, \ldots, J) : t \geq 0)$ interacting with a finite environment process $\mathbf{Y} = (Y(t) : t \geq 0)$ with state space K. Put the "Main assumption" and the "Assumption on the dynamics" and the usual independence assumptions in force.
Assume further that $\eta_j/\mu_j(n) := \eta_j(k)/\mu_j(n,k)$ **is independent of** $k \in K, \forall j = 1, \ldots, J$.
Let $\theta = (\theta(k) : k \in K)$ be the stationary distribution of $\mathbf{Y}$. If $\mathbf{Z} := (\mathbf{X}, \mathbf{Y})$ is ergodic its unique stationary and limiting distribution is for $(n_1, \ldots, n_J; k) \in \mathbb{N}_0^J \times K$

$$\begin{aligned} & \pi(n_1, \ldots, n_J; k) \\ = & \lim_{t\to\infty} P(X(t) = (n_1, \ldots, n_J), Y(t) = k) \\ & = C^{-1} \prod_{j=1}^{J} \prod_{i=1}^{n_j} \left(\frac{\eta_j}{\mu_j(i)} \right) \cdot \theta(k) \,. \end{aligned}$$

Comments: (1) The autonomous environment is not influenced by the evolving network. (2) No mechanism is described which holds $\eta_j/\mu_j(n) := \eta_j(k)/\mu_j(n,k)$ invariant during changes of the environment. (3) Tsitsiashvili et al. [TOKB02] argue with existence of natural control mechanisms of technical and biological systems to react on changes of the environment.

A. *Unreliable networks: Autonomous environment*

We consider Jackson networks where external forces of the environment generate random breakdowns of servers in the network and the repair is also performed under random influences. The environment is of rather general structure: Nodes may break down in isolation or in groups, and batch repair is possible as well.
Usually, when unreliable stations occur, the method of adjusted service rates is used: (1) The reliability of the nodes is computed using methods from reliability theory and yields the percentage of time the nodes are down. (2) The service speeds of the nodes are then decreased by this factor and the performance evaluation methods for product form networks are applied, see [CM96]. The drawback of this method is that the interaction of performance and reliability cannot be studied in a unified model.

Because of the special problem setting we describe the influences of the environment directly by recording the availability of the servers over time. We consider a set $\overline{J} = \{1, \ldots, J\}$ of stations which are unreliable, break down randomly and are repaired thereafter. Repair time is random as well. The "Main assumption" can now be stated as: The availability status of the system can be described by a Markov process $\mathbf{Y} = (Y(t) : t \geq 0)$, where $Y(t) = D$ indicates that at time $t \geq 0$ the stations in $D \subseteq \overline{J}$ are down and under repair, while stations in $\overline{J} \setminus D \subseteq \overline{J}$ are functioning ("are up"). The transition rates (breakdown and repair intensities) of $\mathbf{Y}$ are constructed as follows:

Take any pair of functions $A, B : 2^{\overline{J}} \to [0, \infty)$ subject to $A(\emptyset) = B(\emptyset) = 1$. With these functions define finite breakdown and repair rates

$$Q^{\mathbf{Y}}(D, I) = \frac{A(I)}{A(D)}, \quad D \subset I \subseteq \overline{J}\,,$$

for breakdowns of nodes in non-empty set $I \setminus D$ if nodes in D are already down, and

$$Q^{\mathbf{Y}}(D, H) = \frac{B(D)}{B(H)}, \quad H \subset D \subseteq \overline{J},\,,$$

for finishing repair of nodes in non-empty set $D \setminus H$ if nodes in D are under repair. For all other pairs $G, H \subseteq \overline{J}, G \neq H$, set $Q^{\mathbf{Y}}(G, H) = 0$.

$\mathbf{Y}$ is reversible with respect to the probability measure (with normalization constant $\hat{C}^{-1}$)

$$\hat{\pi} := \left(\hat{\pi}(D) := \hat{C}^{-1} \frac{A(D)}{B(D)}, \quad D \in 2^{\overline{J}} \right). \qquad \text{(III.1)}$$

We fix rules for the network to react to the environment's regime, i.e. we modify the routing matrix r:
(1) Whenever a station is broken down and under repair, service is interrupted and customers present there are frozen, new customers are not admitted to this station.
(2) Therefore we need new routing mechanisms for subsets of functioning nodes. These rules are taken from the literature, e.g., from the area of teletraffic networks with finite buffer stations, resp. from reliability control.

Definition III.2: Rerouting schemes found in the literature which lead to explicit stationary distributions of the network processes are as follows (for details see [SD03][Sections 5, 6] and [DS13][Section 2.3]):

(i) REPEATED SERVICE–RANDOM DESTINATION (RS–RD) ($\equiv$ REFLECTION). This applies only for reversible r, but is then a standard scheme to resolve blocking situations. When a customer tries to visit a down node he is rejected/reflected, stays at his node for another service and tries again with newly selected destination according to r.
(ii) STALLING. Whenever any node breaks down all services and arrivals are interrupted and resumed only when all nodes are up again. This applies in the general setting and stems from reliability control for production of highly dependable systems.
(iii) SKIPPING ($\equiv$ JUMP-OVER-PROTOCOL) is applied in various frameworks to approximate more complicated realistic rerouting schemes. Whenever a customer wants to visit a broken down node he is not allowed to settle down there and has to jump forward according to r until he reaches an up-node or leaves the network.

Proposition III.3: REROUTING SCHEMES IN OPEN NETWORKS. Consider a Jackson network with routing matrix (II.1) and traffic equations for the admitted customers (II.2). When nodes in $D \subseteq \overline{J}$ are down, routing is restricted to $\overline{J}_0 \setminus D$ and determined by a routing matrix $r^D = (r^D(i,j) : i,j \in \overline{J}_0 \setminus D)$ generated by one of the schemes in Definition III.2 The associated traffic equations similar to (II.2) are for $j \in \overline{J} \setminus D$ with $\lambda_j^D := \lambda \cdot r^D(0,j)$

$$\eta_j^D = \lambda_j^D + \sum_{i \in \overline{J} \setminus D} \eta_i^D r^D(i,j), \qquad \text{(III.2)}$$

and are solved by $\eta_j^D = \eta_j, \quad j \in \overline{J} \setminus D, \qquad$ (III.3)

where the η_j are from the solution of (II.2). We set $r^\emptyset := r$ and, if necessary, $\eta_j^\emptyset := \eta_j$.

Definition III.4: The queue length-availability process $\mathbf{Z} = (\mathbf{X}, \mathbf{Y})$ has states $(n_1, n_2, \ldots, n_J, D) = (\underline{n}, D) \in \tilde{\mathbb{E}} = \mathbb{N}_0^J \times 2^{\overline{J}}$. The interpretation: D is the set of servers in *down status*. At node $i \in D$ there are n_i customers waiting for the server being repaired. The strictly positive transition rates for the generator $Q^{\mathbf{Z}} = (Q^{\mathbf{Z}}(\underline{n}, D; \underline{n}', D') : (\underline{n}, D), (\underline{n}', D') \in \tilde{\mathbb{E}}\ )$ of the unreliable Jackson network process are

$$\begin{aligned} Q^{\mathbf{Z}}(\underline{n}, D; \underline{n} + e_j, D) &= \lambda r^D(0,j) \\ Q^{\mathbf{Z}}(\underline{n}, D; \underline{n} - e_i, D) &= \mu_i(n_i) 1_{(n_i>0)} r^D(i,0) \\ Q^{\mathbf{Z}}(\underline{n}, D; \underline{n} - e_i + e_j, D) &= \mu_i(n_i) 1_{(n_i>0)} r^D(i,j) \end{aligned}$$

$$\begin{aligned} Q^{\mathbf{Z}}(\underline{n}, D; \underline{n}, H) &= \frac{B(D)}{B(H)} \qquad H \subset D \\ Q^{\mathbf{Z}}(\underline{n}, D; \underline{n}, I) &= \frac{A(I)}{A(D)} \qquad I \supset D \end{aligned}$$

Theorem III.5: PRODUCT FORM FOR JACKSON NETWORKS WITH BREAKDOWN AND REPAIR. [SD03],[Sau06][Theorem 2.4.1] For any of the rerouting schemes from Definition III.2 holds: If $\mathbf{Z} = (\mathbf{X}, \mathbf{Y})$ is ergodic then the steady state π is with ξ from (II.3) and $\hat{\pi}$ from (III.1) of product form: For $(n_1, \ldots, n_J,\ D) \in \tilde{\mathbb{E}}$ we have $\pi(n_1, n_2, \ldots, n_J,\ D) =$

$$= \hat{C}^{-1} \frac{A(D)}{B(D)} \cdot \prod_{j=1}^{J} \left(C_j^{-1} \prod_{i=1}^{n_j} \frac{\eta_j}{\mu_j(i)} \right). \qquad \text{(III.4)}$$

Here $\eta = (\eta_1, \ldots, \eta_J)$ is the solution of the traffic equation (II.2) when all nodes are up.

Clearly, this formula does not mean that the environment and the queue lengths are independent: The environment is assumed here to be a Markov process for its own, but it strongly influences the service provided by the nodes and even the arrival streams there and, furthermore, the nodes interact as well-the interactions are carried by the traveling customers. These dependencies are not expressed by the one-dimensional (in time) marginal process distribution, which is a multidimensional (in space) product form distribution. In fact, very little is known about the dependence structure of the interacting processes here. A study of what happens to these correlations when changing the environment of the network can be found in [DS13].

B. Non-ergodic networks in a random environment

Goodman and Massey [GM84] studied Jackson networks of single server queues with constant service rates μ_i where at some nodes the stability condition $\eta_i < \mu_i$ is violated. This is related to the topic of this review due to the following observation:
In large and complex networks, instable subnetworks often occur due to sudden changes of the network's environment with local overload or due to non-availability of nodes or links, but nevertheless, there co-exist regions where other subnetworks stabilize locally.

The most recent applications, where nodes or links may be not available for some time, are mobile and ad-hoc networks or sensor networks. The emergence of wireless ad-hoc networks, which are built of varying

sets of mobile users with wireless communication capabilities without relying on a pre-existing infrastructure, introduces problems concerning availability of transmission nodes in the vicinity of a user. On the other hand, in such networks, overload emerges from too many users entering a region and applying for transmission.
Many complex systems require an integrated model to study the interplay of performance and availability in locally instable networks, e.g., with network control and routing protocols in case of link failures in IP networks [MKC03], [NSB+03] or with the handling of catastrophic events in mobile cellular networks [KN03].

Goodman and Massey [GM84] proved that in Jackson networks for the stable nodes $\tilde{S} \subset \overline{J}$ the joint limiting distribution on the stable subnetwork $\tilde{S}$ is exactly of the form (II.3). For local states $(n_i : i \in \tilde{S}) \in \mathbb{N}_0^J$

$$\pi(n_i : i \in \tilde{S}) = \prod_{j \in \tilde{S}} \left(C(j)^{-1} \prod_{k=1}^{n_j} \frac{\eta_j}{\mu_j(k)} \right). \quad \text{(III.5)}$$

We now turn to such networks with unreliable nodes,

Theorem III.6: [MD09][Theorem 15] Consider a Jackson network with unreliable stations with non-ergodic state process $\mathbf{Z} = (\mathbf{Y}, \mathbf{X})$. Let $\eta = (\eta_1, ..., \eta_J)$ denote the unique solution of the generalized traffic equation $\eta = \lambda + (\eta \wedge \mu) \cdot r$ and denote by $\tilde{S} = \{i \mid \eta_i < \mu_i\} \subset \overline{J}$ the subset of stable nodes. We assume that only stations in $\tilde{S}$ break down and have to be repaired. (This restriction is removed in [Myl13][Chapter 5].) Breakdown and repair rates are load-independent as in Section III-A. The rerouting regime is either Stalling, or Skipping, or RS-RD. If RS-RD is in force, we require side constraints which resemble the local balance conditions for reversibility:

$$\eta_i r_{ij} = \eta_j r_{ji} \forall\ i, j \in \tilde{S},\ \eta_i r_{ij} = \mu_j r_{ji} \forall\ i \in \tilde{S}, j \in \tilde{S}^c,$$
$$\mu_i r_{ij} = \mu_j r_{ji} \forall\ i, j \in \tilde{S}^c.$$

Then the joint availability-queue-length process $\mathbf{Z} = (\mathbf{Y}, \mathbf{X})$ is Markov on $\mathbb{N}_0^J \times \mathcal{P}(\tilde{S})$. The limiting behavior of the marginal state distributions is for all $D \subseteq \tilde{S}$ and $n_i, n_j \in \mathbb{N}_0$ as follows :

$$\begin{aligned} &\pi^{\tilde{S}}((n_i, i \in \tilde{S}); D) \\ &:= \lim_{t \to \infty} P\Big((X_i(t) = n_i : i \in \tilde{S}); D\Big) \\ &= \hat{C}^{-1} \frac{A(\bar{I})}{B(\bar{I})} \prod_{i \in \tilde{S}} \left(1 - \frac{\eta_i}{\mu_i}\right) \left(\frac{\eta_i}{\mu_i}\right)^{n_i} \end{aligned}$$

$$\begin{aligned} &\pi^{\tilde{S}^c}((n_j, j \notin \tilde{S}); D) \\ &:= \lim_{t \to \infty} P\Big((X_j(t) = n_j : j \notin \tilde{S}); Y(t) = D\Big) = 0. \end{aligned}$$

For unstable nodes $j \in \tilde{S}^c$ and all $r_j \in \mathbb{N}_+$ it holds

$$\lim_{t \to \infty} P\left(X_j(t) \leq r_j\right) = 0,$$

i.e. the limiting marginal distribution for unstable nodes is the degenerated one-point distribution at ∞.

C. Networks with infinite supply

In many applications to (some or all) nodes is associated a so-called infinite supply, which guarantees that the server's capacity is utilized to the fullest. In classical production control the infinite supply is realized by an inventory from which additional raw material is injected whenever the production facility would become empty. Other applications are e.g. in computer and telecommunications systems.
Wireless sensor networks. The nodes (sensors) continuously sense their environment and have to forward the data to a central station (sink). This is usually not possible by direct communication, so the nodes act additionally as transmission stations for data from other sensor nodes. If forwarding transmissions from other nodes has priority, the own data constitute the infinite buffer which generates the infinite supply for the node.
Computer communication systems. Weiss [Wei05] describes a MAN (metropolitan area network) Ethernet RPR (resilient packet ring), in which ring traffic has priority over the traffic generated at nodes.
Call service centers. There, usually agents, when not answering a call, switch to low priority works such as answering email or administrative duties.

Weiss [Wei05] derived the correct traffic equations for this class of networks and computed under weak additional assumptions the joint limiting distribution for nodes without infinite supply and the local limiting distributions for single nodes with infinite supply. The overall stationary and limiting joint distribution seems to be out of reach. Networks with infinite supply (with some more general features) are embedded into a random environment in [Myl13][Section 4.6]. The environment enforces nodes to break down randomly, being repaired thereafter. The breakdown-repair mechanism is that described in Section III-A. The asymptotic results of Weiss are extended there, see [SDH14] for further contributions.

IV. Non-autonomous environment: Influenced by the network

In many situations the Main assumption that Y is Markov for its own and autonomous is not realistic and therefore the Assumption on the dynamics, with its one-way control ENVIRONMENT $\to$ NETWORK does not meet the problem setting. A first example is the unreliable Jackson network from Section III-A, when the load of the network, i.e. the queue lengths influence the development of the environment. Consequently, the breakdown rates not only depend on the external forces generated by the environment itself, but also react indirectly to the actual load of the server. We will discuss this below in Section IV-A. Thereafter, we turn to a framework in a very different area.

A. Networks with queue length dependent breakdown

The unreliable network from Section III-A is studied with the same state space. The difference is that the transition rates of jumps in the availability coordinate depend on the queue lengths status of the nodes.

Definition IV.1: Take any pair of functions $A, B : \{\emptyset\} \cup \bigcup_{D \in 2^{\overline{J}}} \left(\{D\} \times \mathbb{N}^{|D|}\right) \to [0, \infty)$, define finite breakdown and repair rates as follows for queue length vector $\underline{n} = (n_i : i \in \overline{J})$:

$$Q^{\mathbf{Y}}(D, I; \underline{n}) = \frac{A(I; n_i : i \in I)}{A(D; n_i : i \in D)}, \quad D \subset I \subseteq \overline{J},$$

for breakdowns of nodes in non-empty set $I \setminus D$ if nodes in D are already down, and

$$Q^{\mathbf{Y}}(D, H; \underline{n}) = \frac{B(D; n_i : i \in D)}{B(H; n_i : i \in H)}, \quad H \subset D \subseteq \overline{J}, ,$$

for finishing repair of nodes in non-empty set $D \setminus H$ if nodes in D are under repair. For all other pairs $G, H \subseteq \overline{J}, G \neq H$, set $Q^{\mathbf{Y}}(G, H; \underline{n}) = 0$.

Theorem IV.2: [SD03],[Sau06][Theorem 2.4.1] For any of the rerouting schemes from Definition III.2 and the breakdown-repair regime from IV.1 holds:
If $\mathbf{Z} = (\mathbf{X}, \mathbf{Y})$ is ergodic then the steady state is of product form with normalization constant $\hat{C}^{-1}$:
For $(n_1, \ldots, n_J,\ D) \in \tilde{\mathbb{E}}$ holds $\pi(n_1, n_2, \ldots, n_J,\ D)$

$$= \hat{C}^{-1} \cdot \frac{A(D; n_i : i \in D)}{B(D; n_i : i \in D)} \prod_{j=1}^{J} \left(C_j^{-1} \prod_{i=1}^{n_j} \frac{\eta_j}{\mu_j(i)} \right).$$

$\eta = (\eta_1, \ldots, \eta_J)$ is the solution of the traffic equation (II.2) when all nodes are up.

Example IV.3: Vacation systems. A review on single node queueing systems with server vacations is [Dos90], and [LAP01] describes more recent developments. We can extend this class of models to Jackson networks where empty servers take vacations: Whenever all server in I become idle they wait at most an exponentially-$A(I)$ distributed time for arrival of a new customer. If no customer arrives, the servers take a vacation which is exponentially-$B(I)$. When a vacation is ongoing new customers are rejected. When the vacation expires the returned server again waits an exponential $A(I)$ time for new customers. If new customers arrive, service immediately commences. Otherwise a new vacation is taken by the server, and so on. The function A is then

$$A(I; n_i : i \in I) = A(I) \prod_{i \in \bar{I}} \delta_{0\, n_i} \quad \text{for all } I \subset \overline{J}, I \neq \emptyset,$$

and similarly the function B may take positive values only for (subsets of) nodes which are empty as well:

$$B(I; n_i : i \in I) = B(I) \prod_{i \in I} \delta_{0\, n_i} \quad \text{for all } I \subset \overline{J}, I \neq \emptyset.$$

An issue in network theory is whether a standard (completely reliable) Jackson network can turn into an unstable network if the environment changes, i.e. if in the present framework some nodes become unreliable. The answer is to the affirmative as is shown by an example in [MD09][Proposition 12, Example 13] which elaborate on breakdown and repair rates which are load dependent. On the other side, it is demonstrated in Example 14 there that load independent breakdown and repair rates as in Section III-A cannot destabilize a stable Jackson network.

Economou and Fakinos [EF98] investigated classical product form networks with the additional feature that blocking of service and arrivals can occur due to a rather general regime. They are able to subsume networks with unreliable nodes into this class of models.
They distinguish two kinds of breakdowns: independent and active. A station is subject to independent breakdowns if breakdowns can occur independently of whether service is provided or not, which is the situation from Section III-A, while in the active case a station may break down only when it provides service, that is, when there is at least one job being served in it, which is a framework complementary to Example IV.3.

They represent a queueing network with breakdowns by a mixed network with N + 1 queueing subnetworks. One of these is the original network and describes the service of the jobs. The other N subnetworks are closed networks of two stations with one job each that describe the state of the stations: up or down. For example, for station i the corresponding closed network consists of two stations i-down and i-up and one job. When this job is in i-down, station i is down while when this job is in i-up, station i is up: Saying it the other way round, the N closed subnetworks constitute the environment of the original network.

B. Inventories associated to production systems

We consider a simple production system described by an exponential state dependent single server. To satisfy arriving customers' requests, for each service one piece of raw material is needed, which is taken from an associated inventory. The inventory is controlled by one of the standard replenishment policies of classical inventory theory. If the inventory depletes no new service is possible because of the lack of raw material. We assume that arriving customers finding the inventory empty decide not to join the queue, in terms of inventory theory we observe 'lost sales'.
In the context of this survey we interpret the inventory as the environment which enforces the production server to stop when inventory is empty, and lets the server resume production when next replenishment arrives. There are various versions of such queueing-inventory systems described in the literature, a survey is [KLM11], an early investigation on several models is [SSD$^+$06], an indepth study is [Sch04].

We consider a prototype M/M/1/∞-system with attached inventory and investigate the interplay of inventory management and FCFS queueing regime [SSD$^+$06]:
Customers arrive in a Poisson-λ-stream, service times are exponential-μ. The server needs for each customer exactly one item from the attached inventory. The on-hand inventory decreases by one at the moment of service completion. If inventory is decreased to the reorder point $r \geq 0$, a replenishment order is instantaneously triggered. The replenishment lead times are exponential-ν. The size of the replenishment depends on the inventory management policy. We consider the (r, S)-policy, i.e. replenishment fills the inventory up to maximal inventory size $S > r$.
During the time the inventory is depleted and the server waits for a replenishment order to arrive, no customers are admitted to join the queue ("lost sales").
Let $X(t)$ denote the number of customers present at the server at time $t \geq 0$, either waiting or in service (queue length) and let $Y(t)$ denote the on-hand inventory at time $t \geq 0$. Then $(\mathbf{X}, \mathbf{Y}) = ((X(t), Y(t)), t \geq 0)$, the *queueing-inventory process* is a continuous-time Markov process with state space of $(\mathbf{X}, \mathbf{Y})$ is $E = \{(n, k) : n \in \mathbb{N}_0, k \in K\}$, where $K = \{0, 1, \ldots, S\}$.

Proposition IV.4: [SSD$^+$06] $M/M/1/\infty$ system with (r, S)-policy, $\exp(\nu)$-distributed lead times, and lost sales. The queueing-inventory process $(\mathbf{X}, \mathbf{Y})$ is ergodic iff $\lambda < \mu$. Its steady state distribution $\pi = (\pi(n, k) : (n, k) \in E)$ has product form

$$\pi(n, k) = \left(1 - \frac{\lambda}{\mu}\right) \left(\frac{\lambda}{\nu}\right)^n \theta(k),$$

where $\theta = (\theta(k) : k \in K)$ with normalization constant C is

$$\theta(k) = \begin{cases} C^{-1}(\frac{\lambda}{\nu}) & k = 0, \\ C^{-1}(\frac{\lambda+\nu}{\lambda})^{k-1} & k = 1, ..., r, \\ C^{-1}(\frac{\lambda+\nu}{\lambda})^{r} & k = r+1, ..., S. \end{cases} \quad \text{(IV.1)}$$

An M/M/1/∞-system with (r, Q)-inventory policy, i.e. size of the replenishment order is always $Q > r$, admits similar evaluation [SSD$^+$06]. In this article various queueing-inventory systems are investigated.

Systems with non-exponential lead times are considered in [SAH13] and in [KD15] state-dependent service rates and non-exponential lead times are covered. Although the explicit form of the stationary distributions is more complicated, separability is observed throughout.

A multidimensional (network) system of parallel queues each with an associated inventory under lost sales and a common replenishment Jackson network regulated by base stock policies is investigated in [OKD14]. Separability occurs in various crossover connections and makes performance analysis and inventory optimization of rather complex systems amenable.

A general Jackson network where (some) nodes have an attached inventory is investigated in [SWD07]. The joint inventory process constitutes the environment for

the Jackson network, and it turns out that separability occurs under some additional assumptions.

C. Birth-death-loss processes in a random environment

As will be sketched in Section VIII for birth-death processes in a random environment no separability structures have been found in the classical literature. Only recently a related product form has been proved under the heading "Loss systems in a random environment", where the "loss system" is an $M/M/1/\infty$ queue [KD15][Section 2] or an $M/M/N+1/\infty$ queue [KD15][Section 3] with state dependent service rates, embedded in a general environment K, where bidirectional interaction QUEUE $\leftrightarrow$ ENVIRONMENT governs the development of the system. The principles are:
The queue length dependent service with rates $\mu(n)$ and the Poisson-λ arrival stream of the queue are interrupted from time to time, depending on the status of the environment. To be more precise:
The state space of the environment is the disjoint union of two sets $K := K_B + K_W$ If the environment is in some state inside of $K_B \subset K$ the service and the arrivals are interrupted, i.e. the system is *Blocked*. Whenever the environment returns to $K_W := K \setminus K_B$, service is resumed and new arrivals occur, i.e. the system *Works* again.
The environment changes according to a continuous transition regime $V = (v(k,m) : k,m \in K)$ and is enforced to jump according a one-step Markov matrix $R = (R(k,m) : k,m \in K)$ whenever a customer's service expires and the customer departs.
The non negative transition rates of the Markovian queueing-environment process $\mathbf{Z} := (\mathbf{X}, \mathbf{Y})$ on state space $\mathbb{N}_0 \times K$ are for (n,k)

$$\begin{aligned} q(n,k;n+1,k) &= \lambda, & k \in K_W, \\ q(n,k;n-1,m) &= \mu(n) 1_{(n>0)} R(k,m), & k \in K_W, \\ q(n,k;n,m) &= v(k,m) \in \mathbb{R}_0^+, & k \neq m, \\ q(n,k;i,m) &= 0, \text{ otherwise for } (n,k) \neq (i,m). \end{aligned}$$

The main result is to extract from the system's dynamics a new (reduced) generator matrix Q_{red} of some artificial Markov process on K, which depends on V, R, λ, but not on $\mu(\cdot)$. If this artificial process admits a stationary distribution $\theta = (\theta(k) : k \in K)$ and if the condition

$$C \quad := \quad \sum_{n=0}^{\infty} \left(\prod_{i=0}^{n-1} \frac{\lambda}{\mu(i+1)} \right) < \infty \qquad \text{(IV.2)}$$

holds, then $\mathbf{Z} = (\mathbf{X}, \mathbf{Y})$ is ergodic with product form steady state

$$\pi(n,k) = C^{-1} \left(\prod_{i=0}^{n-1} \frac{\lambda}{\mu(i+1)} \right) \cdot \theta(k). \qquad \text{(IV.3)}$$

An application of this model for analytical investigation of wireless sensor networks (WSN) is developed in [KD14]. Performance analysis of WSN by analytical methods is usually done in two steps: (1) Investigate a single ('referenced') node (RN), and (2) combine by an approximation procedure the interacting nodes to a separable network, a review is [WDW07]. More detailed study of a node model is [Li11], other typical examples for the two-step procedure are [LTL05], [ZL11].

The model and analysis in [WWDL07] is extended in [KD14]. The main idea is to consider the interplay of the message queue (MQ) of RN and its environment. The environment consists of the other nodes of the network plus an internal control structure of RN.
The MQ of RN is modeled as a modified $M/M/1/\infty$ node (the birth-death-loss system above) with standard parameters λ, μ while the environment has complicated structure and is composed of an inner (local) environment of RN and an outer environment. Functioning of RN and its MQ is governed by the following principles which incorporate several interacting processes.
• Length of the MQ (packet queue) of RN,
• number of active nodes in the one-hop neighborhood and the nodal delivery status of these neighbors (the "outer environment"),
• nodal delivery status of the RN (part of the "inner or local environment"),
• modus of RN (active $= 1$, sleep $= 0$) (part of the "inner environment").

RN can communicate with other nodes if and only if RN is **active** $(= 1)$ **and** the number of active nodes in the outer environment is > 0 (for short we say, the outer environment then is **on** (> 0)). This condition determines K_W, and consequently K_B as well. An element $(a, t, \xi, \eta) \in K = K_W + K_B$ reports whether RN is active or asleep (a), the status t of an internal timer of RN, the delivery status ξ of RN, and the combined delivery status η of the nodes in the one-hop neighbourhood of RN. For details see [KD14]. The separability structure of the system is expressed as:
The joint queue length of MQ and environment process

$\mathbf{Z}$ is ergodic on K iff $\lambda < \mu$. Then its unique stationary distribution π fulfills for all $(n, a, t, \xi, \eta) \in \mathbb{N}_0 \times K$

$$\pi(n, a, t, \xi, \eta) = \left(1 - \frac{\lambda}{\mu}\right)\left(\frac{\lambda}{\mu}\right)^n \cdot \theta(a, t, \xi, \eta)\,,$$

where θ is a uniquely determined probability distribution on K.

Several settings of the environment are considered where Θ factorizes further.

V. Non-autonomous environment: Environment dependent capacities and load balancing

In this section we generalize the models of Zhu [Zhu94] and others. They required that the environment is Markov, and if the environment is in state k, parameters are adapted to $\lambda_i(k), \eta_i(k), \mu_i(k)$ (for state independent service rates), and the critical additional assumption is: The utilizations $\eta_i(k)/\mu_i(k)$ do not depend on k, $\eta_i(k)/\mu_i(k) = \eta_i/\mu_i$. Consequently, the local marginal stationary distribution π_i is the same for any environment k.

We will describe an extension of this principle and let moreover the network influence the environment in a way that $\eta_i(k)/\mu_i(k)$ stays invariant during changes of the environment.

This extension will especially resolve the problem that even the simplest networks with autonomous breakdown-repair regime from Section III-A do not fit into the schemes of Zhu et al.: The used rerouting regimes guarantee that the solution η^D of the reduced traffic equations are zero on D, the set of down nodes.

A. Optimal routing and invariance of $\eta_i(k)/\mu_i(k)$

We consider in this section at first a Jackson network as described in Section II and assume that no varying influences of its environment are perceptible (think of a sleeping environment). Conditioned on this and with given load and service capacities we assume that the routing is optimized according to some cost criterion. Such optimization of routing and arriving loads can be found in the literature at many sources. Similarly, optimization of capacities (expressed in the μ_i) to prescribed local load is available. We sketch only a few contibutions:

Optimal design of queueing systems is a challenging topic, a recent survey is the book of Stidham [Sti09]. One of the typical problem structures is as follows: Given the nodes of the network and their service capacities and the external arrival streams (load) to the network, find an "optimal routing" for the customers such that some optimality criterion is satisfied.

In [BS83] a set of stations with fixed service intensities μ_i is located in parallel, and a Poisson arrival stream with overall load λ has to be split and routed to the stations such that a given cost-reward function is maximized. This results in determining the routing of arriving customers and the "optimal utilizations" $\lambda \cdot r(0, i)/\mu_i$ (with self-explaining notation). In [Sti09][Sect. 6.1] in addition the overall load rate λ is subject to control, and in [Sti09][Chapt. 7] the design of general Jackson networks by control of admission and routing is investigated.

With a slightly different setting of the Jackson networks, resp. a more general migration network, described by parameters which are essentially equivalent to ours, Whittle [Whi85][Sect. 2] considered optimal "non-adaptive routing rules" (not depending on the queue lengths as in our model as well). His optimization of routing for fixed external arrival intensities results (in our notation) in optimization of routing probabilities $r(i, j)$. In the cost function in [Whi85] the optimal ratios $\eta_i/\mu_i(n_i)$, resp, $\eta_i^{n_i}/(\prod_{\ell=1}^{n_i} \mu_i(\ell))$ occur.

B. Randomized rerouting

Assume now for the network of Section V that according to some optimization criterion based on the local queue lengths X_j for the fixed $\mu_j(n_j), j \in \overline{J}$, the optimal utilizations η_j/μ_j, resp, the ratios $\eta_j/\mu_j(n_j)$ are determined by an adequate routing scheme r.

We investigate the problem of how to adjust routing when service capacities change, such that the ratios $\eta_j/\mu_j(n_j)$ are maintained. This would guarantee in the sense of Zhu and others that the optimal local queue lengths are maintained optimally.

If the environment awakes and enters state $k \in K$, and if the service intensities at node i are changed by an environment dependent factor $\gamma_i(k) \in [0, \infty)$ for $i \in \overline{J}$: $\mu_i(n_i) \rightsquigarrow \mu_i(n_i) \cdot \gamma_i(k)$, then we have to react in different ways depending on the size of the $\gamma_i(k)$.

(1) It is possible that some nodes break down completely, i.e. $\gamma_\ell(k) = 0$ for such node ℓ. Clearly, the broken down nodes should not be visited any

longer, and from the side constraint to maintain at least approximately the ratios "overall arrival rate"/"service rates" of the other nodes with $\gamma_\ell(k) \neq 0$, it is tempting to try rerouting by modified versions of skipping or reflection.

(2) Indeed, such modifications exist, for a detailed study see [KDO14]. Let us consider first the case $\gamma_i(k) \in [0,1], \forall i \in \overline{J}$, i.e. nodes are only degraded when the environment is in state k. Then the new routing has two components:
• Part of the total external arrival rate will be rejected, and
• the admitted load will be redistributed among the nodes which are not completely broken down in a way to meet exactly the old ratios.
Redistribution will be governed by the following scheduling schemes.

Definition V.1: We start from the rerouting schemes described in Definition III.2 for routing matrix r.
(i) RANDOMIZED REFLECTION (≡REPEATED SERVICE-RANDOM DESTINATION) (RS–RD) WITH REVERSIBLE ROUTING. When a customer tries to visit a down node ℓ, i.e. $\gamma_\ell(k) = 0$, the jump is "accepted" with probability $\alpha_\ell(k) = 0$, i.e. he is rejected, stays at his node for another service and tries again with newly selected destination.
When a customer tries to visit a node ℓ, which is degraded but still up, i.e. $\gamma_\ell(k) \in (0,1]$, the jump is accepted with some positive probability $\alpha_\ell(k)$, and the jump is performed. With probability $1 - \alpha_\ell(k)$ this jump is not accepted, i.e. he is rejected, stays at his previous node for another service and tries again with newly selected destination by r.
(ii) RANDOMIZED SKIPPING. Whenever a customer wants to visit a broken down node ℓ, i.e. $\gamma_\ell(k) = 0$, the jump is "accepted" with probability $\alpha_\ell(k) = 0$, i.e. he is not allowed to settle down there and has to jump forward according to r until he reaches an up-node, possibly degraded, and is accepted there or leaves the network.
When a customer tries to visit a node ℓ, which is degraded but still up, i.e. $\gamma_\ell(k) \in (0,1]$, the jump is accepted with some positive probability $\alpha_\ell(k)$, and the jump is performed. With probability $1 - \alpha_\ell(k)$ this jump is not accepted, i.e. he is not allowed to settle down there and has to jump forward according to r until he reaches an up-node, possibly degraded, and is accepted there or leaves the network.

We will determine the acceptance probabilities $\alpha_\ell(k)$ after describing the other possible modification of service rates due to the environment's influence.

(3) If $\gamma_\ell(k) \in [0,\infty)$, then we either speed up service at node ℓ if $\gamma_\ell(k) > 1$, or have a degraded server at node ℓ if $\gamma_\ell(k) < 1$. When at least one service rate increases, i.e. when $||\boldsymbol{\gamma}(k)||_\infty > 1$, the mechanism to adapt the network's load and routing is:
• We increase the total network input by a factor $\beta(k) = ||\boldsymbol{\gamma}(k)||_\infty > 1$: $\lambda \rightsquigarrow \beta(k) \cdot \lambda$ and
• redistribute the admitted load utilizing either "Randomized reflection" (reversible routing is required) or "Randomized Skipping", with an acceptance probability vector $\boldsymbol{\alpha}(k) = \boldsymbol{\alpha}(\boldsymbol{\gamma}(k))$, see Definition V.1.

(4) Specification of the acceptance probabilities $\boldsymbol{\alpha}(k) = \boldsymbol{\alpha}(\boldsymbol{\gamma}(k)) = (\alpha_\ell(k) : \ell \in \overline{J}_0)$ for environment state $k \in K$:
(i) For an arriving admitted customer his selection of an entrance node is always accepted.
(ii) For a customer in the network his selection of a departure to node 0 is always accepted.
(iii) In case of $\gamma_i(k) \in [0,1], \forall i \in \overline{J}$, i.e. nodes are only degraded when the environment is in state k, set $\alpha_\ell(k) = \gamma_\ell(k)$ for all ℓ.
(iv) For $\gamma_\ell(k) \in [0,\infty)$ with $\gamma_\ell(k) > 1$, for at least one node ℓ, set $\alpha_j(k) = \gamma_j(k)/||\boldsymbol{\gamma}||_\infty$ for all $j \in \overline{J}$.

(5) The randomized rerouting schemes from Definition V.1 define stochastic one-step transition matrices which in any case will be denoted by $r^{\boldsymbol{\alpha}(k)} = (r^{\boldsymbol{\alpha}(k)}(i,j) : i,j \in \overline{J}_0)$. From the context the precise interpretation is always clear.

Remarks: Consider the environment in state $k \in K$.
(a) If $\gamma_j(k) > 1$, node j can process more load without being overloaded, which is easily seen by considering a single $M/M/1/\infty$ queue. In a network however, this additional load departing from j can cause overload at other nodes. Therefore some of the offered new input of rate $\beta(k) \cdot \lambda$ possibly will not be accepted after readjusting the routing. The randomized rerouting algorithms from Definition V.1 will automatically compute rejection rates for the external arrivals at all nodes.
(b) It is not obviuos that in case $\gamma_i(k) \in [0,1], \forall i \in \overline{J}$ we have to choose $\alpha_j(k) = \gamma_j(k)$ to adjust the ac-

ceptance probability of node j precisely to the service rate factor $\gamma_j(k) \in [0,1]$. It means especially that the algorithm automatically detects the correct amount of load which is feasible to maintain the utilizations.

C. Networks with rerouting by randomized skipping

We consider a Jackson network, the development of which is influenced by the time varying status of its system's environment. On the other hand, the network may trigger the environment to change its status. The dynamic is determined on one side by the environment as a jump process $\mathbf{Y} = (Y(t) : t \geq 0)$, changes of which result in changes of the network's parameter, and on the other side by the network process $\mathbf{X} = (X(t) : t \geq 0)$ as jump process where some jumps of $\mathbf{X}$ enforce the environment to immediately react to this jump. To be more precise:

The state of the environment is recorded in a countable environment state space K and whenever the environment at time t is in state $Y(t) = k \in K$ it changes its status to $m \in K$ with rate $v(k,m)$.

The network process $\mathbf{X}$ records the joint queue length vector, as in Section II, and $X_j(t) = n_j \in \mathbb{N}_0$ is the local queue length at node $j \in \overline{J}$. Whenever the environment is in state $k \in K$ and at node j a customer is served and leaves the network, then this jump of the local queue length triggers with probability $R_j(k,m)$ the environment to jump immediately from state k to $m \in K$.

Associated with environment state $k \in K$ is a vector $\boldsymbol{\gamma}(k) \in [0,\infty)^{\overline{J}}$ which determines the factor by which the service capacities are changed, when the environment enters k. This results in a state dependent service rate $\mu_j(n_j,k) = \gamma_j(k) \cdot \mu_j(n_j)$ if the queue length at node j is n_j and the environment is k.

The network reacts to the impact of the environment in state k by modifying the routing according to different strategies, which we have described in Definition V.1.

Additionally, the network may admit more customers, whenever in environment state k there exist some $\gamma_j(k) > 1$. In such state $k \in K$ the overall arrival rate to the network is increased by $\beta(\boldsymbol{\gamma}(k)) = ||\boldsymbol{\gamma}(k)||_\infty$ from λ to $\lambda \cdot \beta(\boldsymbol{\gamma}(k))$.

We emphasize that neither the matrix $V = (v(k,m) : k,m \in K)$, which is (with suitably defined diagonal elements) a generator matrix nor the stochastic matrices $R_j = (R_j(k,m) : k,m \in K), j \in \overline{J}$, need to be irreducible or even ergodic, and furthermore that V is **not** the generator of $\mathbf{Y}$, which in general is not Markov.

We describe here in detail the case of randomized skipping and refer for randomized reflection to [KDO14][Section 4.2]. We need environment dependent rerouting with acceptance probabilities $\boldsymbol{\alpha} = \boldsymbol{\alpha}(\gamma(k))$, modified rerouting matrices $r^{(\boldsymbol{\alpha}(\gamma(k)))}$, and overall load factors $\beta(\gamma(k))$.

To keep notation short we write $\boldsymbol{\alpha}(k) = (\alpha_j(k) : j \in \overline{J}_0)$, instead of $\boldsymbol{\alpha}(\gamma(k))$, $r^{(\boldsymbol{\alpha}(k))}$ instead of $r^{(\boldsymbol{\alpha}(\gamma(k)))}$ and $\beta(k)$ instead of $\beta(\gamma(k))$.

The randomized skipping according to Definition V.1 yields a routing regime $r^{(\boldsymbol{\alpha}(k))}$ and the total service input rate is changed by a factor $\beta(k)$. $\boldsymbol{\alpha}$ and β are defined for $k \in K$ as: $\alpha_0(k) = 1$, and

$$\beta(k) := \begin{cases} 1 & \text{if} \quad ||\gamma(k)||_\infty \leq 1\,, \\ ||\gamma(k)||_\infty & \text{if} \quad ||\gamma(k)||_\infty > 1\,. \end{cases}$$

$$\alpha_j(k) = \begin{cases} \gamma_j(k) & \text{if} \quad ||\boldsymbol{\gamma}(k)||_\infty \leq 1\,, \\ \frac{\gamma_j(k)}{||\boldsymbol{\gamma}(k)||_\infty} & \text{if} \quad ||\boldsymbol{\gamma}(k)||_\infty > 1\,, \end{cases} \quad \forall j \in \overline{J}\,,$$

Under environment condition k we denote $B(\boldsymbol{\gamma}(k))$ and $W(\boldsymbol{\gamma}(k))$ as set of completely down nodes, resp. as set of nodes which, although possibly being degraded or upgraded, can still serve customers.

We denote the coupled process (QUEUE LENGTHS-ENVIRONMENT) by $\mathbf{Z} = (\mathbf{X}, \mathbf{Y}) = (Z(t) : t \geq 0) = ((X(t), Y(t)) : t \geq 0)$ on state space $E := \mathbb{N}_0^J \times K$. The dynamics of $\mathbf{Z}$ relies for the environment process $\mathbf{Y}$ on a generator matrix $V = (v(k,m) : k,m \in K)$ and stochastic matrices $R_j = (R_j(k,m) : k,m \in K), j \in \overline{J}$. Recall that the routing matrix $r = (r(i,j) : i,j \in \overline{J}_0)$ is irreducible. It can be shown that $r^{(\boldsymbol{\alpha}(k))}$ from randomized skipping is irreducible on $W(\gamma(k)) \cup \{0\}$. With usual independence assumptions $\mathbf{Z}$ is Markov on $E := \mathbb{N}_0^J \times K$ with generator $Q^{\mathbf{Z}} = (Q^{\mathbf{Z}}(\underline{n}, k; \underline{n}', k') : (\underline{n}, k), (\underline{n}', k') \in E)$. The strict positive transition rates of $Q^{\mathbf{Z}}$ are for $(\underline{n}, k) = ((n_1, \ldots, n_J), k) \in \mathbb{N}_0^{\overline{J}} \times K$ and $j, i \in \overline{J}$

$$\begin{aligned} Q^{\mathbf{Z}}(\underline{n}, k; \underline{n} + e_i, k) &= \beta(k)\lambda r^{(\boldsymbol{\alpha}(k))}(0,i), \\ Q^{\mathbf{Z}}(\underline{n}, k; \underline{n} - e_j + e_i, k) &= 1_{[n_j > 0]}\gamma_j(k)\mu_j(n_j) \\ &\quad \cdot r^{(\boldsymbol{\alpha}(k))}(j,i),\ i \neq j\,, \\ Q^{\mathbf{Z}}(\underline{n}, k; \underline{n} - e_j, m) &= 1_{[n_j > 0]}\gamma_j(k)\mu_j(n_j) \\ &\quad \cdot r^{(\boldsymbol{\alpha}(k))}(j,0)R_j(k,m), \end{aligned}$$

$$Q^{\mathbf{Z}}(\underline{n},k;\underline{n},m) \quad = \quad v(k,m), m \in K\,.$$

Theorem V.2: Assume the queue lengths-environment process $\mathbf{Z} = (\mathbf{X}, \mathbf{Y})$ is ergodic and assume that the pure Jackson network process $\mathbf{X}$ without environment is ergodic with stationary and limiting distribution ξ for $\underline{n} \in \mathbb{N}_0^{\overline{J}}$

$$\xi(\underline{n}) = \xi(n_1, \ldots, n_J) = \prod_{j=1}^{J} \left(C(j)^{-1} \prod_{k=1}^{n_j} \frac{\eta_j}{\mu_j(k)} \right),$$

with normalizing constants $C(j)$. Define the *reduced generator* Q_{red} as

$$Q_{red} := \left[V + \sum_{j \in \overline{J}} \eta_j I_{(\gamma_j \bullet r^{(\boldsymbol{\alpha}(\cdot))}(j,0))}(R_j - I) \right],$$

where γ_j and $r^{(\boldsymbol{\alpha}(\cdot))}(j,0)$ are for $j \in \overline{J}$ real valued functions on K. Assume the *reduced generator equation* $\theta \cdot Q_{red} = 0$ has a non zero, non negative solution.
Then from ergodicity of $\mathbf{Z}$ follows: Q_{red} is irreducible on K and the *reduced generator equation* $\theta \cdot Q_{red} = 0$ has a strictly positive stochastic solution θ.
Furthermore, the queue lengths-environment process $\mathbf{Z}$ has the unique steady state distribution $\pi = (\pi(\underline{n}, k) : \underline{n} \in \mathbb{N}_0^{\overline{J}}, k \in K)$ of product form:

$$\pi(\underline{n}, k) = \xi(\underline{n})\theta(k), \quad \underline{n} \in \mathbb{N}_0^{\overline{J}},\ k \in K\,.$$

Remark: The reduced generator equation has a stochastic solution if $|K| < \infty$ holds, but there are many other easy to identify cases. The assumption, that $\theta \cdot Q_{red} = 0$ has a non zero, non negative solution, rules out that Q_{red} is transient with only the zero vector as solution of that equation. Similarly, the cases null-recurrent and transient for Q_{red} with non-zero solution are not feasible due to ergodicity of $\mathbf{Z}$.

VI. Random walks in a queueing network environment

Gannon, Pechersky, Suhov, Yambartsev [GPE+15]. introduced an interesting class of models, where the Jackson network acts as the environment for a random walker which circulates on the network's nodes forever. His jump intensities are influenced by the network's state. Such models are investigated already in the celebrated BCMP network models [BCMP75] under the heading of **mixed** networks, i.e. the network serves external customers, entering from the outside and eventually leaving, and a set of fixed internal customers, which cycle in the network forever. The new feature compared to the BCMP framework is that the state of the environment influences the transition intensities of the random walk processes (of the internal customer), creating strong correlations, and vice-versa the position of the random walker influences the transition intensities of the environment (= network) processes. This two-sided interaction RANDOM WALKER $\leftrightarrow$ JACKSONIAN ENVIRONMENT implies that neither the random walk nor the environment process is Markov, as it is required in [Zhu94] and by others. In general this makes analysis much more challenging with respect to technical effort. In [GPE+15] this is diminished by assuming overall reversibility. The simplest model is as follows (notation changed to be in line with the previous sections).

The Jackson network with node set $\overline{J} = \{1, \ldots, J\}$ consists of single server FCFS queues. At all nodes a Poisson-λ stream of indistinguishable customers arrives, service times at all nodes are exponentially μ distributed. Routing inside the network is determined by a symmetric intensity transition matrix

$$B = (\beta(i,j) : i,j \in \overline{J}), \quad \text{with} \quad \beta(i,j) = \beta(j,i).$$

The strictly positive entries of the generator of the joint queue length process $\mathbf{X}$ are for $\underline{n} \in \mathbb{N}_0^J$

$$\begin{aligned} Q^{\mathbf{X}}(\underline{n}, \underline{n} + e_i) &= \lambda, \\ Q^{\mathbf{X}}(\underline{n}, \underline{n} - e_k) &= \mu 1_{(n_k > 0)}, \\ Q^{\mathbf{X}}(\underline{n}, \underline{n} - e_k + n_i) &= \beta(k,i) 1_{(n_k > 0)}, \quad i \neq k. \end{aligned}$$

Into this network a permanently moving (internal) customer is injected (the random walker). The system is described using states $(j, \underline{n})$, where $\underline{n} \in \mathbb{N}_0^J$ records the joint queue lengths of the external customers, while $j \in \overline{J}$ indicates the node where the random walker resides. A Markov process $\mathbf{Z}$ is constructed with state space $\overline{J} \times \mathbb{N}_0^J$ and generator $Q^{\mathbf{Z}}$ for $(j, \underline{n}) \in \overline{J} \times \mathbb{N}_0^J$, with some $\varphi \in \mathbb{R}$, and symmetric matrices

$$\begin{aligned} \Theta &= (\theta(i,j) : i,j \in \overline{J}), \quad \text{with} \quad \theta(i,j) = \theta(j,i), \\ \text{and } T &= (t(i,j) : i,j \in \overline{J}), \quad \text{with} \quad t(i,j) = t(j,i), \end{aligned}$$

$$\text{as} \qquad Q^{\mathbf{Z}}(j, \underline{n}; j, \underline{n} + e_i) = \lambda \cdot e^{\varphi \cdot \delta_{ji}}, \quad \text{(VI.1)}$$

$$Q^{\mathbf{Z}}(j, \underline{n}; j, \underline{n} - e_k) = \mu 1_{(n_k > 0)}, \quad \text{(VI.2)}$$

$$Q^{\mathbf{Z}}(j, \underline{n}; j, \underline{n} - e_k + e_i) = \beta(k,i) 1_{(n_k > 0)},$$

$$j \neq i, k,\ i \neq k, \quad \text{(VI.3)}$$

$$Q^{\mathbf{Z}}(j,\underline{n};j,\underline{n}-e_j+e_i) = \theta(j,i)1_{(n_j>0)}, \quad j\neq i, \tag{VI.4}$$

$$Q^{\mathbf{Z}}(j,\underline{n};j,\underline{n}-e_k+e_j) = \theta(k,j)\cdot e^{\varphi}1_{(n_k>0)}, \quad j\neq k, \tag{VI.5}$$

$$Q^{\mathbf{Z}}(j,\underline{n};\ell,\underline{n}) = e^{\varphi\cdot n_j}\cdot t(j,\ell). \tag{VI.6}$$

The interpretation: Call the node where the random walker resides to be 'loaded', all other nodes are 'unloaded'. Then (VI.1) says that the arrival rate at the loaded node is changed, at unloaded nodes the arrival rate is maintained. (VI.2) indicates that departure rates out of the network are unchanged, similarly transitions of external customers inside the network between unloaded nodes are not changed, see (VI.3).
(VI.4) says that transitions inside the network from the loaded node j to unloaded node i are controlled by the symmetric matrix Θ, while (VI.5) says that transitions inside the network from an unloaded node k to loaded node j are controlled by an e^{φ}-modification of Θ.
Finally, (VI.6) describes the jumps of the random walker from loaded node j to node ℓ, which happens with intensity determined by the matrix T which is modified according to the queue length at the departure node j.

Remark: The Jackson network which is the environment for the random walker is not autonomous due to the dependence of arrival rates on the position of the random walker, see (VI.1). The internal transition rates are influenced by the position of the random walker as well, see (VI.3) - (VI.5).
If we change our point of view, we may consider the random walk as a non-autonomous environment for the Jackson network.

Theorem VI.1: [GPE$^+$15][Theorem 3.1] If $\lambda/\mu<1$ and $(\lambda\cdot e^{\varphi})/\mu<1$ holds, the Markov process $\mathbf{Z}$ with generator $Q^{\mathbf{Z}}$ is ergodic and its unique stationary and limiting distribution is $\pi(j,\underline{n}) =$

$$\frac{1}{J}\left(1-\frac{\lambda\cdot e^{\varphi}}{\mu}\right)\left(1-\frac{\lambda}{\mu}\right)^{J-1}\left(\frac{\lambda}{\mu}\right)^{\sum_{i\in\overline{J}}n_i}\cdot e^{\varphi\cdot n_j}$$

Furthermore, in [GPE$^+$15] embedding of two other systems from statistical mechanics into a Jacksonian environment are investigated: Exclusion processes and zero-range processes, which are closely related to the models in computer and communications systems. Under reversibility these systems defined on finite graphs without environment have product form equilibria.

Most interesting in the context of this review are exclusion processes, which we will describe shortly. An exclusion particle process describes moving particles on a finite graph with set of vertices $\overline{J}=\{1,\dots,J\}$ and resembles a classical Gordon-Newell network if we identify particles with moving customers.
An exclusion process can be described as follows: The node set is $\overline{J}$, at node j the residence times of particles are exponentially distributed with mean μ_j^{-1}. At any node at most one particle can reside (in queueing terms: A single server without waiting room).
There are $N<J$ indistinguishable particles in the system. A particle on leave from node j selects his next destination i with probability $r(j,i)$ according to the **reversible** irreducible stochastic routing matrix

$$r=(r(i,j):i,j\in\overline{J}). \tag{VI.7}$$

If node i is idle, the particle departs immediately and enters i, otherwise it stays on at node j for another exponential-μ_j residence time, and so on.

Proposition VI.2: The exclusion process as described above has state space $F:=\{(k_1,\dots,k_J)\in\{0,1\}^J : k_1+\dots+k_J=N\}$ and the state process $\mathbf{X}=(X(t):t\geq 0)$ where $X(t)=(X_j(t):j\in\overline{J})$ records whether at node j at time t resides a particle. $X_j(t)=0$ indicates that node j is idle, $X_j(t)=1$ indicates a particle residing there.
Denote by $\eta=(\eta_j:j\in\overline{J})$ the probability solution of $\eta=\eta\cdot r$.
Then $\mathbf{X}$ is an ergodic Markov process and its stationary distribution π is for $(k_1,\dots,k_J)\in\{0,1\}^J$

$$\pi(k_1,\dots,k_j)=G(N,J)^{-1}\prod_{i=1}^{J}\left(\frac{\eta_i}{\mu_i}\right)^{k_i}, \tag{VI.8}$$

where $G(N,J)^{-1}$ is the normalization constant.

Comment. The result of Proposition VI.2 is well known in communication networks with finite buffers: If the routing matrix for the messages is reversible and if we apply as blocking resolution protocol *repeated service-random destination* [Onv90][p.502] such a product form equilibrium emerges. This is true even if at the nodes a finite non zero buffer exists [Pit79].

The classical exclusion process from Proposition VI.2 is embedded into an environment. This is done by construction of a Jackson network as described

in Section II which lives simultaneously on the underlying node set $\overline{J}$. The additional assumptions in [GPE$^+$15][Section 4.1] are:
The joint state space is $E = \{0,1\}^J \times \mathbb{N}_0^J$ and the state process records for any node j at time t the number $Y_j(t) = n_j \in \mathbb{N}_0$ of external customers present there and the number $X_j(t) = k_j \in \{0,1\}$ of particles (internal customers) present there. The dynamics of the process $\mathbf{Z} = (\mathbf{X}, \mathbf{Y})$ is driven by transition rates with two-sided interactions. The result: Under strong partial balance conditions a product form equilibrium occurs.

The authors further introduce closed Gordon-Newell networks as environment and open random walk models. In any case considered, they come up with explicit stationary distributions, which are proven by direct insertion into the local balance equations. These are sufficient to be solved because the coupled interacting processes are assumed to be reversible.

VII. Separable models in discrete time

We mention only some examples of discrete time systems in a random environment. Economou [Eco04] provides a general investigation of Markov chains in random environments. In [MD06] a tandem network in a random environment is considered in discrete time. The environment enforces the nodes to break down randomly, similar to the system in Section III-A. For more details on performability metrics for these systems see [Mal08][Chapter 4] and the references there.

VIII. Related models without product form

"Random walks in a random environment" constitute a class of models for the movement of a particle in a disordered irregular medium. These models are used to investigate the asymptotic behaviour of various phenomena in statistical mechanics, biology, and chemistry, to mention only a few applications. The mathematical methods used in this research are almost disjoint to those we elaborated on in the previous sections.
The research sketched in Section VI is on models which resemble systems from the field of "random walks in a random environment", whereas the methods exploited there resemble classical product form calculus.

A survey on movement of a particles in a disordered medium can be found in the lectures on "Random motions in random media" by Sznitman [Szn02], and on "Spin glasses" by Bolthausen [Bol02]; a more recent survey is by Drewitz and Ramirez [DR13].

For single service stations (in isolation) there is a long history with investigations on the behaviour of the stations under external influences, which are subsumed under the term of an environment. Similarly birth-death processes (as generalizations of classical $M/M/1/\infty$ queues) in a random environment are well investigated. Most of this work resulted in complex steady state distributions, see e.g. [CT81], [Cog80], [Yec73], [KY12], [Fal96]. With reference to queueing systems similar investigations are reported in [ZP91] and [PZ95].
Service systems where due to external influences the availability of servers decreases have found much interest in the literature, see e.g. the early paper [Zol66]. A recent survey is [KPC14] ("queues with interruptions"). The results in these articles most often lack the elegance of Jackson's product form steady state and the simplicity of the steady states of birth-death processes.

Further related work is by Yamazaki and Miyazawa [YM95] where the environment of a queue is called the "set of background states" which determine transition rates of the queue and on the other side are influenced by state changes of the queue. Yamazaki and Miyazawa prove a decomposition property (in the spirit of decoupling as in Jackson's theorem) **not** for the queue and the environment but for the joint (QUEUEING-ENVIRONMENT PROCESS) and some supplementary variables which are introduced for Markovisation of the process in case of non exponential service times and non exponential holding times for the background states.

Foss, Shneer, and Tyurlikov [FST12] derived Lyapunow functions for multidimensional processes in an autonomous random environment and applied the obtained conditions to guarantee stability of wireless networks governed by two different protocols. The stationary distribution of the interacting processes seems to be out of reach.

Acknowledgment I thank Bernd E. Wolfinger and Klaus-Dieter Heidtmann for their helpful suggestions to improve the manuscript.

REFERENCES

[BCMP75] F. Baskett, M. Chandy, R. Muntz, and F.G. Palacios. Open, closed and mixed networks of queues with different classes of customers. *J. of the Ass. Comp. Mach.*, 22:248-260, 1975.

[BM13] S. Balsamo and A. Marin. Separable solutions for Markov processes in random environments. *European J. of Oper. Res.*, 229(2):391-403, 2013.

[Bol02] E. Bolthausen. Lectures on spin glasses. In E. Bolthausen and A.-S. Sznitman, eds. *Ten Lectures on Random Media*, *DMV Seminar* vol.32, chapt.2, 55-109. Birkhäuser, Basel, 2002.

[BS83] C. E. Bell and S. Stidham Jr. Individual versus social optimization in the allocation of customers to alternative servers. *Management Science*, 29:831-839, 1983.

[CM96] R. Chakka and I. Mitrani. Approximate solutions for open networks with breakdowns and repairs. In Kelly. F.P., Zachary. S., and I. Ziedins, eds, *Stochastic networks, Theory and Applications*, chapt.16, 267-280. Clarendon Press, Oxford, 1996.

[Cog80] R. Cogburn. Markov chains in random environments: The case of Markovian environments. *Ann. Prob.*, 8(5):908–916, 1980.

[CT81] R. Cogburn and W. C. Torrez. Markov chains in random environments: The case of Markovian environments. *J.Appl. Prob.*, 18:19-30, 1981.

[Dij88] N. M. van Dijk. On Jackson's product form with 'jump-over' blocking. *OR Letters*, 7(5):233–235, 1988.

[Dos90] B.T. Doshi. Single server queues with vacations. In H. Takagi, ed, *Stochastic Analysis of Computer and Communication Systems*, 217-267, Amsterdam, 1990. IFIP, North-Holland.

[DR13] A. Drewitz and A. F. Ramirez. Selected topics in random walks in a random environment. Techn.Rep. 2013.

[DS08] H. Daduna and R. Szekli. Impact of routeing on correlation strength in stationary queueing networks processes. *J.Appl. Prob.*, 45:846-878, 2008.

[DS13] H. Daduna and R. Szekli. Correlation formulas for markovian network processes in a random environment. Preprint 2013-05, Math. Stat. und Stoch. Proz., FB Mathematik Universität Hamburg, 2013. to appear in: Adv. Appl. Prob., 2015.

[Eco04] A. Economou. Stationary distributions of discrete-time Markov chains in random environment: Exact computations and bounds. *Stochastic Models*, 20:103-127, 2004.

[Eco03] A. Economou. A characterization of product-form stationary distributions for queueing systems in random environment. *Int. J. of Simulation*, 4:4-11, 2003. Proc. of the 17th European Simulation Multiconf.

[Eco05] A. Economou. Generalized product-form stationary distributions for Markov chains in random environments with queueing applications. *Adv. Appl. Prob.*, 37(1):185-211, 2005.

[EF98] A. Economou and D. Fakinos. Product form stationary distributions for queueing networks with blocking and rerouting. *Queueing Syst.*, 30:251-260, 1998.

[Fal96] G. Falin. A heterogeneous blocking system in a random environment. *J. Appl. Prob.*, 33:211 – 216, 1996.

[FST12] S. Foss, S. Shneer, and A. Tyurlikov. Stability of a Markov-modulated Markov chain, with application to a wireless network governed by two protocols. *Stochastic Systems*, 2(1):208–231, 2012.

[GM84] J. B. Goodman and W. A. Massey. The non–ergodic Jackson network. *J. Appl. Prob.*, 21:860-869, 1984.

[GPE+15] M. Gannon, Pechersky, E., , Y. Suhov, and V Yambartsev. Random walks in a queueing network environment. Techn. rep., Version 3: arXiv:1410.1460v3, 2015; to appear: J. Appl. Prob.

[Jac57] J.R. Jackson. Networks of waiting lines. *Oper. Res.*, 5:518-521, 1957.

[KD14] R. Krenzler and H. Daduna. Modeling and performance analysis of a node in fault tolerant wireless sensor networks. In K. Fischbach and U.R. Krieger, eds, *Measurement, Modelling, and Evaluation of Computing Systems and Dependability and Fault-Tolerance*, 73–78, Heidelberg, 2014. GI/ITG, Springer.

[KD15] R. Krenzler and H. Daduna. Loss systems in a random environment - steady state analysis. *Queueing Syst.*, 2015. to appear.

[KDO14] R. Krenzler, H. Daduna, and S. Otten. Randomization for Markov chains with applications to networks in a random environment. Prepr. 2014-02, Math. Stat. und Stoch. Proc., Deptm. Math., Univ of Hamburg, 2014.

[KLM11] A. Krishnamoorthy, B. Lakshmy, and R. Manikandan. A survey on inventory models with positive service time. *OPSEARCH*, 48:153–169, 2011.

[KN03] K. Kordybach and S. Nousiainen. Radio resource management in WCDMA-based networks in emergency situations. In J. Charzinski, R. Lehnert, and P. Tran-Gia, eds, *Providing Quality of Service in Heterogeneous Environments*, *Teletraffic Science and Engineering*, vol.5a, 331-338, Amsterdam, 2003. Proc. of the 18th Int. Teletraffic Congr., Elsevier.

[KPC14] A. Krishnamoorthy, P.K. Pramod, and S.R. Chakravarthy. Queues with interruptions: a survey. *TOP*, 22:290–320, 2014.

[KY12] V. Kulkarni and K. Yan. Production-inventory systems in stochastic environment and stochastic lead times. *Queueing Systems*, 70:207-231, 2012.

[LAP01] H.W. Lee, B.Y. Ahn, and N.I. Park. Decompositions of the queue length distributions in the MAP/G/1 queue under multiple and single vacations with N-policy. *Stochastic Models*, 17:157–190, 2001.

[Li11] W.W. Li. Several characteristics of active/sleep model

in wireless sensor networks. In *New Technologies, Mobility and Security (NTMS)*, pages 1–5, 2011.

[LTL05] J. Liu and T. Tong Lee. A framework for performance modeling of wireless sensor networks. *IEEE Int. Conf. on Communications 2005*, vol.2, 1075-1081, 2005.

[Mal08] C. Malchin. *Stochastic networks: Discrete and continuous time models*. PhD thesis, Depm. Math., University of Hamburg, 2008.

[MD06] C. Malchin and H. Daduna. Availability and performance analysis in a discrete time tandem network with product form steady state. In R. German and A. Heindl, editors, *Proc. GI/ITG Conf. on Measuring, Modelling and Evaluation of Computer and Communications Systems*, 381-398, Berlin, 2006. VDE-Verlag.

[MD09] J. Mylosz and H. Daduna. On the behavior of stable subnetworks in non-ergodic networks with unreliable nodes. *Computer Networks*, 53(8):1249–1263, 2009.

[MKC03] M. Menth, S. Kopf, and J. Charzinski. Impact of resilience requirements on the performance of network admission control methods. Techn. rep. 309, Univ. of Würzburg, Inst. Computer Sci., 2003.

[Myl13] J. Mylosz. *Local Stabilization of Non-Ergodic Jackson Networks with Unreliable Nodes*. PhD thesis, Univ. Hamburg, Dept. Math., 2013.

[NSB+03] A. Nucci, B. Schroeder, S. Bhattacharyya, N. Taft, and C. Diot. IGP link weight assignment for transient link failures. In J. Charzinski, R. Lehnert, and P. Tran-Gia, eds, *Providing Quality of Service in Heterogeneous Environments*, *Teletraffic Sci. and Eng.* vol.5a, 321-330, Amsterdam, 2003. Proc.18th Int. Teletraffic Congr., Elsevier.

[OKD14] S. Otten, R. Krenzler, and H. Daduna. Integrated models for production-inventory systems. Prep. 2014-01, Math.Stat. und Stoch. Proz, FB Mat. Universität Hamburg, 2014. submitted.

[Onv90] R. O. Onvural. Closed queueing networks with blocking. In H. Takagi, ed, *Stochastic Analysis of Computer and Communication Systems*, 499-528. North-Holland, Amsterdam, 1990.

[Pit79] B. Pittel. Closed exponential networks of queues with saturation: the Jackson-type stationary distribution and its asymptotic analysis. *Math. of OR*, 4:357-378, 1979.

[PZ95] N.U. Prabhu and Y. Zhu. Markov-modulated queueing systems. *Queueing Systems*, 19(4):449, 1995.

[SAH13] M. Saffari, S. Asmussen, and R. Haji. The M/M/1 queue with inventory, lost sale and general lead times. *Queueing Systems*, 75:65–77, 2013.

[Sau06] C. Sauer. *Stochastic product form networks with unreliable nodes: Analysis of performance and availability*. PhD thesis, Univ. Hamburg, Dept. Math., 2006.

[Sch04] M. Schwarz. *Stochastic Models in Inventory Theory*. Shaker Verlag, Aachen, 2004.

[SD03] C. Sauer and H. Daduna. Availability formulas and performance measures for separable degradable networks. *Economic Quality Control*, 18:165–194, 2003.

[SDH14] J. Sommer, H. Daduna, and B. Heidergott. Jackson networks with infinite supply and unreliable nodes. Preprint 2014-03, Math. Stat. und Stoch. Proz., FB Math. Univ. Hamburg, 2014. submitted.

[SSD+06] M. Schwarz, C. Sauer, H. Daduna, R. Kulik, and R. Szekli. M/M/1 queueing systems with inventory. *Queueing Systems*, 54:55-78, 2006.

[Sti09] S. Stidham Jr. *Optimal design of queueing systems*. CRC Preess, Boca Raton, 2009.

[SWD07] M. Schwarz, C. Wichelhaus, and H. Daduna. Product form models for queueing networks with an inventory. *Stochastik Models*, 23:627-663, 2007.

[Szn02] A.-S. Sznitman. Lectures on random motion in random media. In E. Bolthausen and A.-S. Sznitman, eds, *Ten Lectures on Random Media*, vol. 32 of *DMV Seminar*, chapter 1,3-51. Birkhäuser, Basel, 2002.

[TOKB02] G.S. Tsitsiashvili, M.A. Osipova, N.V. Koliev, and D. Baum. A product theorem for Markov chains with application to PF-queueing networks. Prepr., Univ. of Trier, Dept. IV, Trier, Germany, 2002.

[WDW07] Y. Wang, H. Dang, and H. H. Wu. A survey on analytic studies of Delay-Tolerant Mobile Sensor Networks. *IEEE Trans. Wireless Communications*, 6(9):3287-3296, 2007.

[Wei05] G. Weiss. Jackson-networks with unlimited supply of work. *J. Appl. Prob.*, 42:879-882, 2005.

[Whi85] P. Whittle. Scheduling and characterization problems for stochastic networks. *J. Royal Stat. Soc., Ser. B*, 47(3):407-415, 1985.

[WWDL07] H. Wu, H., Y. Wang, H. Dang, and F. Lin. Analytic, Simulation, and Empirical Evaluation of Delay/Fault-Tolerant Mobile Sensor Networks. *IEEE Trans. on Wireless Communications*, 6(9):3287-3296, 2007.

[Yec73] U. Yechiali. A queuing-type birth-and-death process defined on a continuous-time markov chain. *Oper. Res.*, 21(2): 604-609, 1973.

[YM95] G. Yamazaki and M. Miyazawa. Decomposability in queues with background states. *Queueing Systems*, 20:453-469, 1995.

[Zhu94] Y. Zhu. Markovian queueing networks in a random environment. *OR Letters*, 15:11 – 17, 1994.

[ZL11] Y. Zhang and W. Li. An energy-based stochastic model for wireless sensor networks. *Wireless Sensor Networks*, 3(9):322–328, 2011.

[Zol66] V. M. Zolotarev. Distribution of queue length and number of operating lines in a system of Erlang type with random breakage and restoration of lines. *Selected Translations in Math. Stat. and Prob.*, 6:89–9, 1966.

[ZP91] Y. Zhu and N.U. Prabhu. Markov-modulated PH/G/1 queueing systems. *Queueing Systems*, 9(3):313-322, 1991.

Novel Methodologies for Real-Time Performance Evaluation of NFV-Based Network Infrastructures

Paul J. Kuehn
Institute of Communication Networks and Computer Engineering (IKR)
University of Stuttgart
E-Mail: paul.j.kuehn@ikr.uni-stuttgart.de

***Abstract*—The current developments in NFV (Network Function Virtualization) and SDN (Software Defined Networks) are mainly directed to functional aspects of future Service Delivery Platforms (SDP) and network infrastructures and need to be complemented by non-functional aspects such as reliability, security/privacy, or Quality of Service (QoS). This contribution aims at novel methods of modeling for performance evaluation as a basis for control/protocol function definitions and Service Level Agreement (SLA) guarantees. For these new architectures two problems direct to new challenges: (1) Real-Time Performance of Cloud Data Centers (CDC) as data and service function repositories and (2) Control plane processes for SDN and signaling for the communication between network controllers. In this paper, two novel approaches will be presented: Load-Based Control algorithms for a minimized power consumption within CDCs based on a novel automatic server consolidation under real-time SLA constraints, and queuing networks with synchronization control constraints for the communication between distributed network controllers. The first problem is solved by Finite-State-Machine (FSM)-controlled multi-server queuing systems which can be mathematically analyzed by a novel recursive algorithm under Markovian conditions. The second problem is largely unsolved for general stochastic service processes; in the past such problems have been approached mostly by Stochastic Petri Net simulations. In this paper an analysis method is presented which is based on a novel methodology for exact task graph reductions under generally distributed stochastic task durations developed initially for the exact performance analysis of parallel processes. Applications of both methodologies, which have been published recently [5,7], are currently in progress.**

Keywords*—cloud data centers, automatic server consolidation, FSM-controlled queuing system, SDN control plane processes, parallel processing, task graph reduction.

1. Introduction

The Future Internet develops towards a Service Delivery Platform (SDP) based on heterogeneous infrastructures of legacy and innovative technologies. SDN, NFV and Cloud concepts will be the enabling methodologies and technologies to provide the flexibility and efficiency for the transformation of the current infrastructure together with new technologies to provide all types of services of a future SDP. In this transformation process, some existing paradigms have to be changed to meet the requirements of real-time performance and Service Level Agreements (SLA), not only in the data transport plane but also in the control plane where virtualization, software-defined functionalities and cloud-based data and function repositories are the key enabling technologies. Real-time services dominate already the existing Internet which are partly supported by a resource over-provisioning but cannot meet more severe real-time SLAs and raise challenges for the provision of higher non-functional criteria for SDPs.

Data Centers consisting of thousands of servers and huge storage capacities are extremely power-hungry, not only through the electronic heat waste but also for cooling; they contribute significantly to the current carbon footprint. Server consolidation, load-dependent activation/deactivations, sleep modes and Dynamic Voltage and Frequency Scaling (DVFS) are the key approaches to reduce the electric power consumption; these methods pay-off as such servers are only utilized by about 30 %. Techniques for power reducing operations of CDCs are subject to DC Management Operation and should be automated to operate dynamically dependent on the current load; such techniques are currently in use, but little has been reported about their real-time performance nor in context with prescribed SLAs as in case when they are part of the Future Internet control plane. Our approach in this direction differs as we suggest a method where the active server configuration and power saving operations are derived from given SLA requirements: the SLA is expressed quantitatively by the average or even by the percentile of the response times of DB access requests with respect to those requests which are delayed, i.e., these requirements can be directly used in context with guarantees on the Quality of Experience (QoE).

In Section 2 a model for the automatic server consolidation, hot and cold stand-by server operation, sleep modes of idled servers, as well as DVFS is suggested and modeled as a FSM-controlled multi-server queuing system. The model is based on a multiple parallel hysteresis which is perfectly suitable for control software implementation or for model simulations. The model is analytically solved by a multi-dimensional Markov Chain (MC); the typical “state explosion” problem for the MC approach or the rigorous mathematical analysis arising with the seminal approaches based on Green's Function applied to half-lattice Markov Chains by Ibe and Keilson [10] and on stochastic complementation by Liu and Golubchik [11], respectively, is avoided through a novel recursive algorithm which provides numerical performance results even for very large parameter values far beyond the feasibilities of the pure analytical methods as in [10, 11].

In Section 3 a novel methodology will be presented which has been developed initially for the exact performance analysis of parallel processing based on task graph representations for

each computational process [7]. In that contribution two more novel results have been found: (1) A generalization of Amdahl's famous Law for the efficiency improvement by parallel computation as well as on the real-time performance of parallel computation, and (2) on the neutrality of parallel processing compared to serial processing with respect to the energy consumption. This methodology can be applied likewise also to queuing networks with synchronization constraints as they appear in communication applications, e.g., communication protocols with ARQ (Automatic Repeat Request) and Timeout control, or to control plane signaling processes between distributed network controllers. Such problems have been modeled and treated in the past only by simulations of Stochastic Petri Net (SPN) models or numerically by mapping the SPN to a multidimensional Markov Chain which had to be solved numerically, but which suffers typically from the "state space explosion" problem and the assumption of memoryless stochastic processes.

Current control in the Internet is limited to end-to-end transport protocols, individual packet forwarding, un- scalable Integrated Services or relative Differentiated Services protocols, and traffic conditioning. Virtualization is used in Virtual LAN (VLAN) and Virtual Private Network (VPN) concepts. More efficient packet forwarding can be achieved by the method of Multi- Protocol Label Switching (MPLS) based on the establishment of a data plane MPLS-path without any control plane support. Mobility within the fixed part of the Internet is supported by Mobile-IP and similar other protocols. The simultaneous transition of the IPv4-based Internet to IPv6 allows further functions of service support by, e.g., extended header concepts, but is still subjected to the individual packet forwarding within the IP layer instead of a control plane support. With the OpenFlow concept, SDN, and Cloud-based NFV platforms a control plane can be established which supports the transition of the current Internet to the Future Internet, new service creations in a rather flexible manner, the establishment of MPLS- paths under SLA conditions, the configuration of VLANs and VPNs, a better manageability of the network with respect to resource utilization, routing control, and fault/reliability/availability/ performance management.

Principal concepts of SDN and NFV have been studied intensively by research projects for about one decade. Many experimental testbed results have shown their usefulness and realizability. Major challenges are now standardization of architectures, interfaces, protocols and data structures as a basis for development, deployment, and interoperability of equipment. In parallel to these developments theoretical studies and prototyped setups have to be performed and experimentally verified with respect to feasibility, planning support, resource dimensioning and network operation. Various standardization activities can be observed currently as the ETSI Network Function Virtualization with respect to Infrastructure Architectural Frameworks, Virtual Network Function Architectures, and Service Quality Metrics [1]. Specifically, ETSI defines a Forwarding Graph method to describe the interconnection between distributed NFV Functions. The IETF suggests a Service Function Chaining method for Data Centers, Mobile Networks, and Long-Lived Flows [2]. The Open Network Foundation defines use-cases for Optical Transport, Carrier Ethernet Network Virtualization and Packet Optical Integration [3]. The W3C Consortium and IETF have developed an Open Standard Framework WebRTC for interactive services based on Browser communications [4]. In our contribution we will mainly follow the ETSI Forwarding Graph approach which can be treated by task graph methods similarly as for parallel processing with synchronization constraints.

2. Load- and SLA-Based Automatic Control of Power-Saving Data Center Operation

Cloud-based Data Centers are increasingly applied for storage and delivery of data, but can also be applied for delivery/support of functions in various configurations as public, private or hybrid "Cloud". The economy of these applications is obvious through a better resource sharing, higher reliability and security, and fast delivery by high-speed communications. At the other hand, Data Centers require a huge amount of energy for their processing/storage devices and cooling, i.e., they are subjected themselves to a power-saving operation ("greening"). The biggest power consumption is required by processing devices and can be largely reduced by the method of server consolidation, i.e., by adapting the number of activated processing devices to the actual load, still under meeting negotiated SLAs based on virtualization of tenants' job requirements and an automatic mapping of virtual machines (VM) on physical processing devices. Consolidation can be executed by deactivating of servers either by switching-off or by low-power sleep mode operation.

A model has been developed which describes server configurations for individual VMs or of a whole Data Center by multi-server service systems which are controlled through a Finite State Machine (FSM) by monitoring the development of load arrivals and service completions continuously, and where decisions about activation of idled/sleeping servers or deactivations of processing modules are derived automatically by an adaptive multiple- parallel hysteresis mechanism which is parameterized by a given SLA either with respect to average job response time or even with respect to percentiles of the response time distribution function [5,6]. The models are analyzed exactly under Markovian traffic conditions by a novel recursion algorithm which allows the extension of the basic on/off-switching of servers to server sleep modes including overhead times for booting deactivated servers/warm-up of sleeping servers as well as server speed throttling by Dynamic Voltage and Frequency Dcaling (DVFS). Analysis results have been reported on the real-time performance of the proposed Data Center model, see [5,6].

In Fig. 1 the basic structure of the queuing model is shown for a two-dimensional state space (x,z) for x activated servers and z queued jobs with server consolidation acc. to a parallel hysteresis state transition model for arriving rate λ for new jobs and service rate μ. Fig. 2 shows the corresponding state-transition diagram (STD). This STD is 3-dimensional: the blanc states refer to states where no server is in an activation (i.e., booting or in a wake-up phase), while the shaded states represent states here servers are in the activation phase.

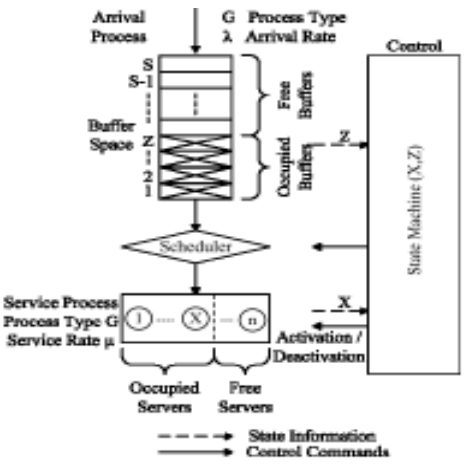


Fig. 1. General Queuing Model for DC Servers, Buffers aand Control

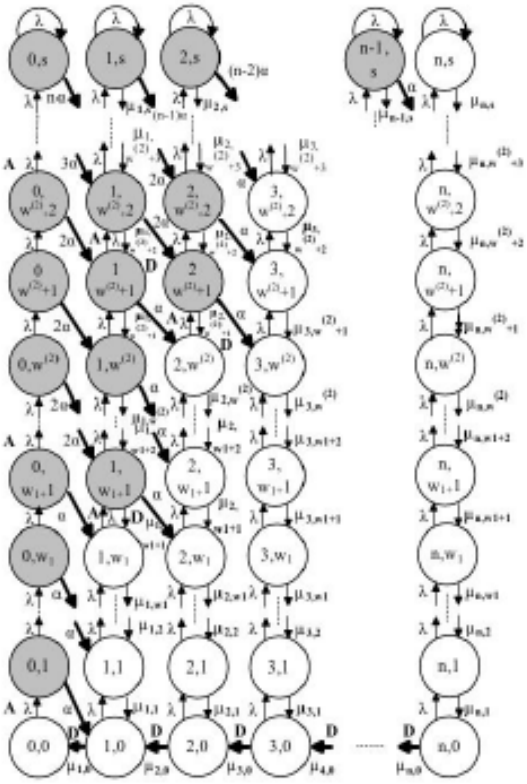


Fig. 2. State Transition Diagram for Multiple Parallel Hystereses, and Threshold-based Activations with Overhead

The difference lies only in different rates α for activation (cold standby mode) or warm-up (hot stand-by mode). Sleeping states refer to idle servers; they consume some small energy while sleeping compared to the switch-off state where no energy is consumed. DVFS is modeled by a state-dependent service rate as lowering voltage supply or clock frequency result in smaller processing speed, i.e., smaller service termination rates. Transition rates between states result from new job arrivals and service terminations; the highlighted transition arrows indicate server activations or deactivations while the thin transitions indicate changes in the buffer space. A new server activation takes place only when the buffer size crosses a certain threshold; this threshold can be set such to meet a prescribed SLA value for the average or percentile of the response time.

Performance results on the server activation rates, on the mean response time/ percentiles of response times, and on the power consumption efficiencies are reported in [5] and [6], respectively. Among others, the new algorithm for the automatic server consolidation results in a scalable SLA adaptation, especially it could be shown that the average response times are insensitive over a wide load range between very low and close-to-saturation load factors. These results lead to the conjecture that the automatic load-based server consolidation is especially advantageous to apply it to real-time CDC for SDN and NFV applications for the Future Internet.

3. Distributed NFV Function Control by Forwarding Task Graph Reduction Methods

For distributed NFV control a fast exchange of control information is required between distant controllers, e.g., in order to find and to set up an MPLS-path (Multiple Protocol Label Switching) across the Internet automatically upon the header analysis at the edge router. The result of this control plane activity could be the reservation of the whole path and either to download the forwarding table entrances for label swapping in all trans-passed routers or to apply the label/port-correspondences for all trans-passed routers in an extended IPv6 packet header for direct execution on each packet in each trans-passed router. This control action uses the transport capability of the underlying packet network, but is subjected to real-time requirements for an efficient establishment and use of the MPLS forwarding capabilities. For this example and many other application cases several partner instances in the control plane are involved where the control execution time depends not only on individual response times but also on synchronization conditions .

The problem defined by the ETSI Forwarding Graph method [2] can be modeled by a queuing network with synchronization constraints which resembles parallel processing of computer programs and will, therefore, be modeled by Task Graphs [7]. For the analysis of such generalized queuing networks only one method is known by modeling as a Stochastic Petri Net, analyzed by simulation (or, theoretically, by transforming it into a multi-dimensional Markov Chain under Markovian statistical conditions). In [7] a novel method has been developed which allows an exact stepwise reduction of Task Graphs under general stochastic conditions into a single virtualized task. The method has been implemented in a tool for the analytical execution [8] as well as as an extension to the general SimLib tool of IKR[9].

To characterize the new method of task graph reductions for parallel computation consider a generic Task Graph example for a program out of [7], shown in Fig. 3.

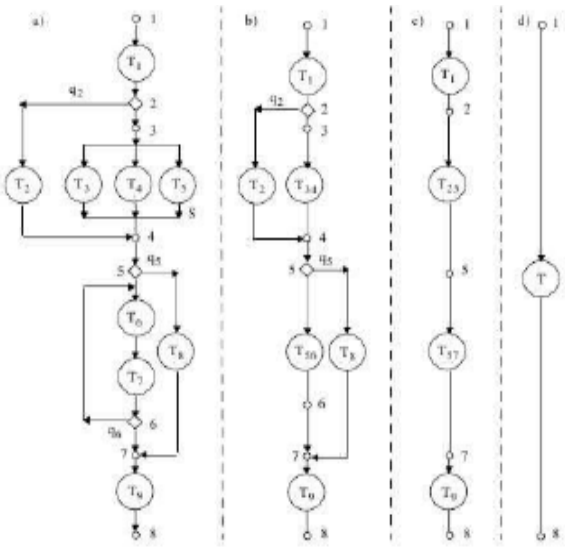


Fig. 3 Task Graph Example and its Stepwise Reduction

The Task Graph in part a) reflects the 4 principal program structure elements of structured programming, c.f. Fig. 4.

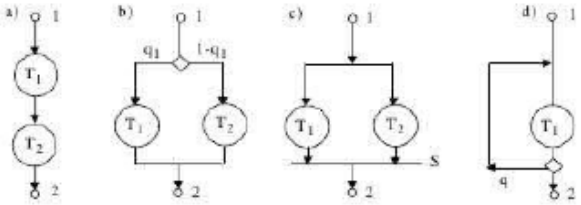


Fig. 4. Principal Program Structure Elements

a) Serial Processing (Concatenation)

b) Random Splitting of Execution Path

c) Parallel Processing with Synchronization Point S

d) Repetition (Iteration, either fixed or probabilistic)

Figs. 3 b, 3c, 3d show a stepwise reduction of the Task Graph by combinations according to the four elementary structure elements shown in Fig. 4, resulting finally in a **single virtualized task.** Such reductions should be applicable for any Acyclic Directed Graph and any program which is guided by the structured programming paradigm. We describe the task processing times in general by arbitrary (general) stochastic probability density functions (which might be known from monitoring of program executions). Under these assumptions we can reduce each of the 4 principal structure elements exactly by probabilistic operations: convolution, random split, maximum (in case of parallel program executions) or minimum (in case of search operations as in Big Data Analytics), and random or constant number of repeated concatenations. The results are density functions of each of the four elementary structure elements. These operations can be performed exactly in principle; for programming reasons we perform these operations exactly for a phase-type model of a task, represented by a series of a constant phase and a degenerated hyper-exponential phase, by which we can model any mean value and any coefficient of variation c between 0 and infinity. The application of these operations leads finally to the first and second moment of the single virtualized phase in Fig. 3d. The four basic operations have been implemented by a program tool which can be applied to any Task Graph under the above stated assumptions [8]. We have also implemented parallel computations for queuing systems for the IKR simulation Tool SimLib where each job which has to be executed on a multi-processor system is executed according to its accompanied very specific Task Graph [9]. The analytic method has also been verified against simulations which showed a precise agreement.

The conclusions of the summarized and outlined Task Graph Reduction method are:

1. Given a multi-processor or multi-core computer system and a representation of programs by stochastic task graphs, the performance can be reduced to solve a **single-server** queuing model of the type GI/G/1, i.e. for a general input process of type GI (General Independent) and a virtual processing time of the type G as result of the Task Graph reduction methodology. For a multi-core system with a given number of cores where only one job is executed at a time for all possibilities of parallel program executions (Note: they can be larger or smaller than the number of cores !).

2. The method leads also to a generalization of Amdahl's Law, as Amdahl assumes that all programs can utilize the full parallel execution capability of the system, while the real parallelisms follows from the specific Task Graph of a job independently of the computer configuration.

3. It has also been shown that for the above assumption of 2. parallel computation is only faster than serial computation for small load factors. For larger load factors parallel computation causes empty processor phases when the degree of parallelism is smaller than the number of processors/cores; this causes so-called “slack times” which lead to a reduction of the maximum utilization of the server system. As a consequence, serial computation (which does not suffer from slack times) has a larger capacity limit and outperforms parallel computation; this effect seems to be largely misunderstood in the literature and is especially important for real-time processing in overload situations.

For distributed NFV control a fast exchange of control information is required between distant controllers, e.g., in order to find and to set up an MPLS path across the Internet automatically upon the header analysis at the edge router. The result of this control plane activity could be the reservation of the whole path by to downloading the forwarding table entrances for label swapping in all trans-passed routers of a flow or to apply the label/port - correspondences for all trans-passed routers in an extended IPv6 packet header for direct execution on each packet in each trans-passed router. This control action uses the transport capability of the underlying packet network, but is subjected to real-time requirements for an efficient establishment and use of the MPLS forwarding capabilities. For this example and many other application cases several partner instances in the control plane are involved where the control execution time depends not only on individual response times but also on synchronization conditions. The problem defined by the ETSI Forwarding

Graph method can be modeled by a queuing network with synchronization constraints which resembles parallel processing of computer programs and will, therefore, be modeled by Task Graphs [7].

The application of the Task Graph reduction method to NFV and SDN- based applications is currently under study. First examples will be presented on ongoing research of distributed NFV function control and on the exact analysis of timer-controlled ARQ protocols under real-time conditions.

Acknowledgements

The contributions to this research by Ms. Maggie Ezzat Mashaly of the German University in Cairo (GUC), Sebastian Scholz of the IKR institute and of the two Master Students Fangli Pi and Felix Beck are greatly appreciated.

References

[1] ETSI GS NFV Documents on Network Function Virtualization.www.etsi.org/technologies-clusters/technologies/nfv

[2] IETF Documents on Service Function Chaining-Data Centers, Mobile Networks, Long-Lived Flows. IETF drafts, July 2014.

[3] Open Network Foundation Documents on Optical Network Use Cases, Carrier Ethernet Network Virtualization, 2014.

[4] WebRTC. Open Standard Framework. W3C Consortium and IETF on WebRTC.

[5] Paul J. Kuehn, Maggie Mashaly, Automatic Energy Efficiency Management of Data Center Resources by Load-Dependent Server Activation and Sleep Modes. 2nd Conf.on Energy Efficiency Data Centers (E^2DC), Berkeley, May 21, 2013. Revised and Extended Version published in: Elsevier Journal Ad Hoc Networks 25 (2015), pp. 497 - 504.

[6] M. Mashaly, P.J. Kuehn, Modeling and Analysis of Virtualized Multi-Service Cloud Data Centers with Automatic Server Consolidation and Prescribed Service Level Agreements, 23 rd ICCTA, Alexandria, Egypt, Oct. 29 - 31, 2013.

[7] P.J. Kuehn, Performance and Energy-Efficiency of Parallel Processing in Data Center Environments, 3rd Conf. On Energy Efficient Data Centers (E^2DC), Cambridge/UK, June 10, 2014. Revised and Extended Version published in: Springer LNCS, No. 8945, pp. 17 – 33, 2015.

[8] Fangli Pi, Analysis of Parallel Computation Processes in Multi- Core/Multi-Processor Systems by Task Graph Reductions. Master Thesis, University of Stuttgart/IKR,2014.

[9] Felix Beck, Analysis of Parallel Computation Processes in Multi-Core/Multi-Processor Systems by Simulation.Master Thesis, University of Stuttgart/IKR, 2015.

[10) O.C. Ibe, J. Keilson, Multi-Server Threshold Queues with Hysteresis. J. Performance Evaluation 21 (1995),pp. 185 – 213.

[11] J.C.S. Lui,L. Golubchik, Stochastic Complement Analysis of Multi-Server Threshold Queues with Hysteresis. J. Performance Evaluation 35 (1999), pp. 119 – 48.

Back to the Future: Towards an Internet of People (IoP)

Peter Reichl
Research Group Cooperative Systems
University of Vienna
Vienna, Austria
peter.reichl@univie.ac.at

Andrea Passarella
Institute for Informatics and Telematics (IIT)
National Research Council (CNR)
Pisa, Italy
a.passarella@iit.cnr.it

***Abstract*— While the recent transition from the network-focused concept of Quality of Service (QoS) to the user-centric notion of Quality of Experience (QoE) has led to a fundamental rethinking of the concept of service quality in communication systems. In this position paper we argue for a much broader understanding of the underlying paradigm change, i.e. with respect to economic and business aspects as well as the perspective of cognitive and social sciences, thus contrasting the current hype around the Internet of Things (IoT) with our vision of (re-)emphasizing an "Internet of People (IoP)". In order to highlight some of the consequences, we will discuss convergence aspects of cyber-physical systems together with recent advances in QoE research, especially concerning the economics of user-perceived quality. For further illustration we will specifically touch upon some results from a fixed point model of QoE-aware charging, before finally focusing on the recently proposed novel concept of Experience Level Agreements (ELA) which is supposed to complement traditional Service Level Agreements (SLA) if it comes to selling QoE to the end user.**

Keywords—IoT; IoP; Quality of Experience; Service Level Agreements; Cyber-physical Systems

I. INTRODUCTION

Since the first steps of the ARPANET in the early 1970's, the Internet has run through an almost breath-taking evolution, establishing itself as the central communication infrastructure of our planet. Today, the Internet leads political revolutions and lets financial markets tremble to an extent almost unthinkable of until few years ago, it has fundamentally changed our communication behavior, last not least due to social media like Twitter, Facebook or WhatsApp, it produces virtually every day a new major uproar around some network or data security issues, it enables and undermines our access to the essential asset of objective and trustworthy information at the same time, and it has started to shift our entire social, cultural and creative life more and more into virtual spaces.

While from this list it is immediately evident that the Internet is (and has always been) a system primarily serving the communication needs of humans and society, it has also to be acknowledged that, after all, it is a mere technical artifact in the sense of a neutral communication infrastructure. Thus, its sole purpose consists of sending data packets from one endpoint to the other, which at the very end breaks down to being just a (rather sophisticated system) of host addresses together with their efficient resolution. As an inevitable consequence, this has led to the fundamental dichotomy of highly specialized technology experts devising and implementing a system for the broad public which, in extremis, is sometimes even considered to be largely composed of the infamous species of "Dumbest Assumable Users" (DAU).

Of course, it can be argued that such a dichotomy is more or less characteristic for any large scale engineering undertaking. However, in the case of the Internet, it is the sheer size of the chasm which is remarkable, originating both from the accomplished degree of technological sophistication and the extremely broad relevance of the Internet as the all-embracing communication infrastructure of the world of today. The resulting consequences concern the widely discussed problem of "digital divide" as far as the different ways of Internet usage are concerned. Moreover, on a much more fundamental level, at the same time they apply also to the definition and formulation of key concepts and notions linking the technical nature of the Internet to the human nature of its users.

As an important illustration, consider the discussion of how to define an appropriate notion for "quality" in the Internet, and specifically the recent paradigm change from technology-oriented "Quality of Service (QoS)" to user-centric "Quality of Experience (QoE)". Here, it is interesting to note that the importance of defining quality along the (subjective) end user needs originally had shaped much of the initial definition of related concepts, especially the definition of QoS. However, this was largely neglected during almost two subsequent decades of corresponding technology-driven research, before eventually being re-discovered a few years ago and finally leading to the establishment of new quality models which today are able to take both the subjective and the objective side into appropriate account [10].

In this position paper, we would like to argue that this kind of paradigm change, integrating economic and business aspects as well as the perspective of cognitive and social sciences into what originally used to be a pure engineering endeavor, will become more and more important for the future global evolution of the Internet. This concerns especially the current hype around the so-called "Internet of Things (IoT)" which aims at integrating the physical world into computer-based systems, resulting in visions like the "smart refrigerator"

automatically refilling itself once empty, etc. Note that we do not intend to question the importance of IoT research in general, however, we would like to argue that the primary purpose of the Internet should still consist of providing communication facilities for humans rather than physical objects.

We call this alternative vision the "Internet of People (IoP)", in order to underline its user-centric characteristics. It will require distinct approaches for a range of fundamental concepts, and thus might exhibit significant differences compared to the emerging IoT. On the other hand, we strongly believe that the Internet should be kept a uniform system, and recent advances in the area of cyber-physical convergence indeed indicate promising directions towards bridging the gap between the connected physical world and the human user.

The remainder of the paper is structured as follows: in section 2, we will briefly review the concept of Cyber-Physical Convergence and link it to the framework of communication ecosystems. Section 3 starts with an introduction to QoE and then focuses on recent advances in the area of Economics of QoE which illustrate important aspects of the envisaged transition towards the IoP. Section 4 summarizes some key arguments of the paper and provides a concluding outlook.

II. Cyber-physical Convergence and Anti-Copernican Revolution

A. Revisiting Cyber-Physical Convergence

The diffusion of personal (mobile) devices and pervasive communication technologies is expected to exponentially increase in the next few years (for example, CISCO foresees a tenfold increase of mobile data traffic between 2014 and 2019, with a compound annual growth rate (CAGR) of 57% [1]). This is increasingly pushing the vision of Cyber-Physical Convergence, discussed for instance in [3] and [4]. According to this vision, the physical world of the users and the cyber world of Internet applications and services are more and more integrated and converging. Data generated in the physical world (e.g., by sensors embedded in personal users' devices and physical infrastructures) flows to the cyber world, where it is elaborated and exchanged. On the other hand, interactions in the cyber world are resulting in actions in the physical world (e.g., because users modify their behaviour based on information received through Internet applications, or because physical infrastructures are configured through actuators).

B. Towards an Anti-Copernican Revolution in ICT

As a key effect of this convergence, humans are more and more placed at the centre of the technical systems they use. Humans and the cyber systems through which they communicate become actors of a complex socio-technical ecosystem [9], and designing effective communication systems needs to take into consideration human behaviour as a structural paradigm, rather than as an afterthought. In [11], this paradigm change has been termed the "Anti-Copernican Revolution", as it puts (back) the human being at the centre of the design and evaluation processes of future communication systems.

According to this communication ecosystem view, we perceive future research on Internet-based communication systems as a truly inter-disciplinary task, shaped by at least four main interacting dimensions. More precisely: this ecosystem perspective links our traditional technology-oriented perspective closely to social, economic and cognitive sciences (describing the behaviour of humans) for designing the communication and data exchange mechanisms of the future Internet. Here, ICT technologies provide the basic enabling solutions for communication to occur. However, the algorithms and protocols for communication and data exchange are not driven exclusively by the need to optimise network resource usage (as in the design of legacy Internet systems). In the converged cyber-physical environment, user devices become proxies of their users in the cyber world: they communicate, exchange and manage data by "emulating" the way their human users would do if interacting with each other in the physical world. Social sciences model the way users establish social relationships, how they trust each other, and how they are prepared to share resources with each other. Communication systems exploiting these models ("social-aware networking protocols") have proved to be very efficient in supporting communication in human-centred mobile networks [4][5]. Cognitive psychology describes, among others, how human beings perceive and interact with data, how they assess relevance of information, how they exchange it when interacting, and how they extract knowledge out of it. Data-centric communication systems for mobile networks have already been proposed, where these models are exploited to efficiently guide information diffusion among users [2]. Micro-economics, last not least, is modelling how humans negotiate the use of infrastructure and content resources, trade and share them. This is also fundamental knowledge to predict how they can interact with each other through communication systems, and to embed such knowledge in the design of the corresponding systems.

We would like to stress the fact that the proposed human-centric approach to the design of future Internet-based communication systems is not yet another bio-inspired networking design wave. Because of the fact that user devices act as proxies of their users, embedding efficient models of human behaviour in the core design of communication systems is a natural way to make devices behave as their human users would do if faced with the same choices and decisions. Moreover, this approach is not confined to designing human-centred applications. Instead, the inter-disciplinary approach of the IoP impacts all conventional layers of the communication stack above the enabling communication technologies, and brings advantage at all layers, as shown by the mentioned examples.

C. Resulting Paradigms

The IoP approach can be the basis for a seamless communication ecosystem for Cyber-Physical Convergence, where communication entities can be humans, their personal devices, as well as other "machines" communicating in the cyber world. Specifically, we can foresee at least three corresponding classes of future communication paradigms:

- *The "Human proxy" paradigm*: This is based primarily on communications between devices, whereby the users' personal devices communicate with each other acting as proxies of their human users.

- *The "Crowdsourcing" paradigm*: This is based both on device-to-device as well as on device-to-human and human-to-human interactions. The human user is perceived as another entity of the communication ecosystem, whose behaviour can be modelled and predicted (clearly, up to a certain extent), with resources that can be exploited to optimise the operations of the system. As an example, we may think of crowdsourcing systems, where humans are used to solve complex problems in a synergic way together with computers.
- *The "User experience" paradigm*: This is based on interactions between users and devices, and the behaviour of the devices is designed taking into consideration the reactions of the human users to the service offered by the communication system, and the resulting quality of the users' experience [9].

In this sense, Quality of Experience (QoE) may be viewed as another cornerstone for the design of communication systems which are following the Cyber-Physical Convegence framework. We will discuss this in the following section in more detail, focusing on the economics of QoE as an especially interesting interdisciplinary aspect.

III. The Example of Quality of Experience

A. From QoS to QoE

Since the early days of Internet research, the question of how to guarantee or at least reliably measure the quality of delivered services has been of paramount interest both to academia and industry. Thus, the concept of Quality of Service (QoS) has been broadly established already in the 1990's, resulting in huge research efforts as well as a plethora of related publications investigating all possible aspects of network parameters like bandwidth, delay, jitter, packet loss rates etc. On the other hand, interestingly enough, most attempts to implement such concepts in productive networks have not been very successful (like, e.g., the IntServ and DiffServ architectures, to mention just two notorious cases). This remarkable failure has been attributed, amongst others, to a certain lack of interrelation between differentiated quality and differentiated pricing [6], which directly underlines the necessity of including the microeconomic point of view into the overall picture. Moreover, the integration of further interdisciplinary perspectives, especially those of cognitive and social sciences, recently has led to an interesting paradigm change from QoS towards the subjective concept of Quality of Experience, i.e. the "degree of delight of the user of a service" [8] depending on content, network, device, application, user expectations/goals and context of use [8].

Note that, in terms of research methodology, QoE research is strictly based on user trials (subjective testing) where typically so-called MOS (Mean Opinion Score) values are determined in order to reflect the average user's judgement on the quality of the offered services. In contrast, mapping QoS parameters to corresponding levels of user satisfaction is considered rather an (often valuable) approximation of true QoE metrics, even if the relationship between QoS and QoE is a recurrent issue while formulating appropriate QoE models, see e.g. [8] or [12]. Moreover, current models also focus on "QoE influence factors" originating from the physical, temporal, economic, task, social, technical and information context.

Hence, QoE basically models the interaction between humans and ICT services through a human-centric approach, taking into consideration human expectations on the quality to be obtained, as well as reactions to varying level of quality. Thus, QoE models can be fruitfully integrated in the communication systems design, for example to anticipate the effect of devices behaviour on the human users, or to understand how users could react and behave when exposed to certain tasks to be carried out in collaboration with devices.

B. Economics of Quality of Experience

As already mentioned previously, there is a very natural close link between service differentiation and corresponding differentiated tariff models, independently of whether we are talking about QoS or QoE. However, there is an important subtlety to be observed if it comes to QoE-enabled charging, as here a differentiated tariff structure constitutes also a part of the user context, and as such gives rise to a direct feedback cycle within the QoE evaluation process. This important fact can be easily illustrated as follows: suppose an end user is signing a contract with a network provider in order to get access to the Internet, however under a tariff which obliges the user to pay high fees. Then, by definition, QoE evaluation results for this user will be lower than for another end user who has been signing a contract with lower charges (if the QoS level stays the same), simply because of the different expectations towards the quality of the delivered service: the user who is paying much is also expecting much, and vice versa.

The corresponding formal model boils down to a fixed point problem which has been discussed in detail in [13]. The key result can be summarized as follows: in a network with dynamic pricing where increasing user demand for limited bandwidth causes QoS degradation which as a consequence lowers charges (and vice versa), under rather mild conditions two (trivial) fixed points exist: a stable one where low user demand allows for excellent QoS at a high price (premium service), and an unstable one where high user demand leads to low QoS at zero cost (free best effort service). As soon as QoE comes into play, both these trivial fixed points break down, and a unique non-trivial fixed point appears instead which is stable, essentially determined by the tradeoff between tariff and corresponding expected QoE on the user side.

It is interesting to note that this formal model has been validated through a series of user trials described in [14] and further analyzed in [13]. In these experiments, users have been offered streaming video at around 20 different quality (and pricing) levels together with the opportunity to choose the quality/price combination which they liked most. As main results, it turned out that users indeed are willing to pay for quality (note that, for the experiment, users have been given real money which they could, but were not obliged to, spend for watching videos in better quality). Moreover, most of the users approach their final choice of quality/price along a convergence pattern closely resembling a damped sine wave, which is precisely what the formal model of [13] is predicting.

C. *From Service Level Agreements (SLA) to Experience Level Agreements (ELA)*

The evolution towards an IoP has of course also direct impact on the way network providers and their customers will interact in the future, for instance if it comes to selling QoE to the users. Currently, there is already a well-established mechanism for arranging binding service-related contracts between providers and users based on clearly specified QoS parameters: the so-called Service Level Agreement (SLA). Unfortunately, SLAs do not foresee explicitly to take the user-related QoE aspects into proper account. Hence, in [15] a novel concept called "Experience Level Agreement (ELA)" has been proposed, whose basic characteristic may be easily summarized as "what SLA is for QoS, ELA will be for QoE". On a more formal level, an ELA can be defined "as a special type of SLA designed to establish a common understanding of the quality levels that the customer will experience through the use of the service, in terms that are clearly understandable to the customer and to which he or she can relate" [15]. This definition underlines already that SLA and ELA have much in common (and in fact will eventually have to coexist), while ELAs allow to additionally meet several QoE-specific challenges like monitoring and validation of subjective service quality as well as marketing issues or the question of how to handle transit service agreements from an end-to-end QoE perspective (for a more detailed discussion please refer to [15]).

IV. Conclusions and Outlook

From its very beginning, the Internet has been conceived as an infrastructure enabling communication for human users, while at the same time being characterized by a high and ever increasing degree of technical sophistication. Over time, this has led to a continuous interplay of conflicting forces as well as various attempts to resolve the resulting tensions. However, the current hype around the Internet of Things (IoT) might turn out to become a real game changer as it explicitly focuses on the communication between elements of the physical world only. Whereas this evolution for sure has its advantages, in this paper we argue that it might require specific efforts to keep the future Internet also an "Internet of People" (IoP). While, thus, the ideas behind the IoP are not new at all, and indeed may be safely considered to be just a step "back to the future", at the same time the need for related paradigm changes is evident, as demonstrated in the case of the successful transition from QoS to QoE. In addition, the concept of Cyber-Physical Convergence as discussed earlier will provide a fundamental framework for eventually bringing both worlds together in a unifying holistic approach.

Above all, it is important to note that the discussion around the IoP is just about to start, hence the present paper does in no way claim providing any final answers or even any sort of completeness, instead its sole purpose is the attempt to trigger further discussion. On the other side, as the notion of IoP is already starting to enter also the public debate [7], it seems high time for the scientific community to start working on it also from a networking point of view.

Acknowledgments

The authors would like to thank their colleagues from the IFIP TC6 Strategic Board and from Dagstuhl Seminars 09192, 12181 and 15022 for many helpful discussions. Further inspiration originates, amongst others, from a range of several projects funded by the European Commission, including FP7 SOCIALNETS (2008-11), FP7 RECOGNITION (2010-13), and FP7 ETICS (2010-13), as well as COST Action IC1003 (QualiNet, 2010-14) and the CELTICplus project QuEEN (Quality of Experience Estimators in Networks, 2011-14).

References

[1] Cisco, "Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update 2014–2019," Cisco, Tech. Rep., http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-etworking/indexvni/white_paper_c11-520862.html, 2015.

[2] M. Conti, M. Mordacchini, and A. Passarella, "Design and performance evaluation of data dissemination systems for opportunistic networks based on cognitive heuristics," ACM Trans. Auton. Adapt. Syst., vol. 8, no. 3, pp. 12:1–12:32, 2013.

[3] M. Conti, S. K. Das, C. Bisdikian, M. Kumar, L. M. Ni, A. Passarella, G. Roussos, G. Troster, G. Tsudik, and F. Zambonelli, "Looking ahead in pervasive computing: Challenges and opportunities in the era of cyber–physical convergence," Pervasive and Mobile Computing, vol. 8, no. 1, pp. 2–21, 2012

[4] E. Daly and M. Haahr, "Social network analysis for information flow in disconnected Delay-Tolerant MANETs," IEEE Trans. Mobile Comput., vol. 8, no. 5, pp. 606–621, May. 2009.

[5] P. Hui, J. Crowcroft, and E. Yoneki, "Bubble rap: Social-based forwarding in delay tolerant networks," IEEE Trans. Mobile Comput., vol. 10, no. 11, pp. 1576–1589, Nov. 2011.

[6] R. Jain, "Internet 3.0: Ten Problems with Current Internet Architecture and Solutions for the Next Generation", Proc. IEEE MILCOM'06, pp. 1-9, October 2006.

[7] J. Judge and J. Powles, "Forget the internet of things – we need an internet of people". The Guardian, May 25, 2015.

[8] S. Möller, A. Raake (eds.), "Quality of Experience: Advanced Concepts, Applications and Methods". Springer, 2014.

[9] P. Reichl, "Quality of Experience in Convergent Communication Ecosystems", in: A. Lugmayr, C. Dal Zotto (eds.): The Media Convergence Handbook. Springer, Sept. 2015.

[10] P. Reichl: "It's the Ecosystem, Stupid: Lessons from an Anti-Copernican Revolution of User-centric Service Quality in Telecommunications". 6th International Conference on Developments in e-Systems Engineering (DeSE'13), Abu Dhabi, United Arabic Emirates, December 2013.

[11] P. Reichl, "From QoS to QoE: Buzzword Issue or Anti-Copernican Revolution?" Keynote abstract, EuroNF Workshop on Traffic Management and Traffic Engineering for the Future Internet, Dec. 2009.

[12] P. Reichl, S. Egger, S. Möller, K. Kilkki, M. Fiedler, T. Hossfeld, C. Tsiaras, A. Asrese, "Towards a Comprehensive Framework for QoE and User Behavior Modelling". Proc. QoMEX 2015, Costa Navarino, Messinia, Greece, May 2015.

[13] P. Reichl, P. Maillé, P. Zwickl, A. Sackl: A Fixed-Point Model for QoE-based Charging. Proc. ACM SIGCOMM 2013 Workshop on Future Human-Centric Multimedia Networking, pp. 33-38, Hong Kong, China, August 2013.

[14] A. Sackl, P. Zwickl, S. Egger, P. Reichl, "The QoE Alchemy: Turning Quality into Money. Experiences with a Refined Methodology for the Evaluation of Willingness-to-Pay for Service Quality", Proc. of 4th International Workshop on Quality of Multimedia Experience (QoMEX'12), Yarra Valley, Australia, July 2012

[15] M. Varela, P. Zwickl, P. Reichl, M. Xie, H. Schulzrinne, "Experience Level Agreements (ELA): The Challenges of Selling QoE to the User". Proc. IEEE ICC 2015–QoE-FI Workshop, London, UK, June 2015.

Performance Exploration of Software-based Packet Processing Systems

Daniel Raumer, Florian Wohlfart, Dominik Scholz, Paul Emmerich, and Georg Carle

Technische Universität München, Department of Computer Science, Network Architectures and Services
{raumer|wohlfart|scholzd|emmericp|carle}@net.in.tum.de

***Abstract*—Software systems that are placed on commodity hardware are integral components of today's networks. They gain momentum as an attractive alternative to dedicated hardware routers, switches, and firewalls. Their big advantages are the high customizability of the software and increased cost-efficiency of the necessary hardware. To improve the performance, features intended to speed up the processing, are steadily added. For assessing influences on the performance, the software and its interaction with available hardware features have to be tested and modelled. Therefore, we evaluate the performance of the Linux Network Stack in different use cases and develop a performance prediction model. Using both black- and white-box measurements the internal behaviour of the Linux router is analysed when approaching data rates up to 10 Gbit/s and the impact of occurring performance limiting factors is studied. External measurements are restricted to the packet rate, while white-box measurements are performed using the profiling tool `perf`. From the results of these measurements, the CPU cycles required to process each packet and their distribution to the tasks executed by the involved driver and kernel functions is analysed.**

***Index Terms*—measurement, software-based networked systems, performance correlation, performance model**

I. Introduction

Through steady development the performance of commodity hardware has been continuously increased in recent years. Especially multi-core architectures have therefore become more interesting to fulfil network tasks by using software packet processing systems instead of dedicated hardware. In particular, software routers using UNIX-based operating systems (OS) are attractive, as a fully functional protocol stack that handles the processing, in this case forwarding on layer 3 of the ISO/OSI model, is implemented. The advantages of high flexibility and lowered costs are contrasted with possible higher performance and reduced energy consumption achieved with specialized hardware [1], [2]. Furthermore, for software routers, features can be rapidly deployed or modified, whereas dedicated hardware would consume extensive development cycles before being able to be upgraded.

With the continuous growth of the internet, the demands on the used techniques increase, too. While 1 GbE is common in home networks, 10 GbE and 40 GbE is common standard in carrier grade networks. Even 100 GbE is already standardised [3]. While hardware routers are able to accomplish these line rates, software routers reach their limit [1]. However, software routers are in general capable of reaching the same performance that hardware routers deliver today [1], [4]. Problems arise especially for small packet sizes: the full line rate cannot be reached as the maximum number of packets per second that can be processed is limited by the CPU. Other factors are further limiting the performance of a software router [1], [4], [5].

While new techniques improve the performance, the interaction with other mechanisms has to be understood and analysed to prevent unwanted behaviour or possible side effects. A common approach is black-box testing: the software router is tested in a certain scenario, using varying input traffic, and, for instance, the resulting packet rate is used to compare the results. Although this shows how many cycles per packet are needed to process a packet, it is not further analysed where exactly, in relation to which function of the OS, the performance is decreased.

In this paper we present a bottom-up performance prediction model of the components involved in packet forwarding on LINUX-based packet processing systems. It is intended to show which functions of the kernel are involved to accomplish each general task and which share of the CPU cycles per packet is consumed by each of them. We use our model for white-box tests, to explain the router internals and analyse the influence of performance limiting factors. Furthermore, different use cases can be compared in respect to which task consumes more or less cycles and, therefore, is responsible for performance losses or gains. This includes measurements with different router configurations and different types of traffic.

Another common test is to confront the router with different packet rates to evaluate the dynamic behaviour of the kernel. Most of the time these measurements are only carried out up to the point at which the CPU is under full load. Although this is understandable as it is not normal for a system to work continuously under overload, forcing the router under overload over a longer period of time can be caused by misconfiguration or by an attack, targeting the CPU of the software router.

By maintaining generality the model also applies to other forwarding frameworks. While the general steps that are necessary to forward packets remain the same for most frameworks, different approaches can be compared.

The remainder of this paper is structured as follows. Based on a discussion of related work about performance exploration of (Linux) software based packet processing in Section II, we describe the internals of packet processing in Linux-based

software switches and routers, in Section III. In Section IV, we discuss the components that are relevant for the performance breakdown and formulate relations as analytical models for software based packet processing systems. Section V utilizes our models in a case study, for a detailed exploration of the performance in which we discuss the used measurement methodology (in Section V-A). We end with a summary of our contributions in Section VI.

II. Software-based Packet Processing Systems

Wu and Crawford [6] gave an in-depth description of which components are involved and what each of them does when receiving a packet that is destined for an application in user space, already in 2006. To analyse the performance they developed a mathematical model of the whole process: the NIC is modelled using the token bucket algorithm, in which tokens are represented by packet descriptors, and all other processing steps are modelled as queueing processes. They come to the conclusion that, aside from the CPU, the size of the *rx_ring* (packer reception buffer) influences the bottleneck as packets can be dropped if not enough descriptors are available due to "memory pressure".

Bolla and Bruschi [5], [7] showed a deep understanding of the processing steps. They used RFC 2544 compliant test cases to analyse the performance of different hardware and data plane architectures, focusing on the Linux kernel version 2.6 [5]. They used throughput, latency and profiling measurements to evaluate an optimized Linux 2.6 kernel, the Click Modular router and the SMP Linux 2.6 kernel. According to their results, they obtained rates up to 2.5 Gbit/s with an optimized kernel version. Today, the used kernel versions underwent changes in comparison to versions used these days. Anyway, the underlying analysis of the packet processing steps, ranging from the New API (NAPI) to the transmitting NIC, contributed to our performance prediction model. Especially the interaction of interrupts and poll-rate are interesting as it is still valid with today's performance optimized NIC drivers (cf. [8]). Bolla and Bruschi stated the fundamental that *the CPU limits the number of packet headers that can be processed*, while bandwidth and latency of the I/O busses limit the total throughput [7]. Furthermore, they show based on their measurements and analysis that shared data structures like the *tx_ring* of the NIC cause extra CPU cycles, as the access to those is serialized using mutex-like techniques. In addition, the data has to be synchronized between the caches of multiple CPUs causing more overhead. Based on these results they propose a multi-core packet forwarding software architecture. The evaluation shows that they are able to reach a packet rate up to 4 Mpps.

In 2009, Dobrescu et al. [4] revisited software router architectures and performed black-box tests. They were already close to reach the 10 Gbit/s line rate when using software routers with multiple 1 Gbit/s ports and minimally sized packets. Studies from 2013 showed that the trend towards continuously increased performance continued [1]. Beifuß et al. [8] presented a study of the packet reception process and developed a simulation model of it. Emmerich et al. [9] systematically described the bottlenecks concerning hardware and software, which we kept in mind during the design of our model.

Besides general purpose Network IO Software, frameworks that focus on high-speed packet processing have been developed: e.g. netmap [10], Intel DPDK [11], or PF_RING ZC [12]. While these are designed to fulfil general network tasks, they are also able to function as a software router or switch, e.g. DPDK vSwitch [13], Click on netmap [14], and VALE [15]. We also designed our model in conformance with these approaches to packet processing. For an in depth analysis of high speed network IO frameworks we refer to our recent publication and papers referenced in there [16].

III. General Steps of Packet Processing

Our abstract model of packet processing is shown in Figure 1: The NIC at which a packet arrives (1) has to fulfil two tasks. Firstly, the packet is transferred to the main memory of the system (2). A descriptor (pointer) to the memory region is then stored in a queue (3), to keep track of arrived packets. Secondly, the NIC has to inform the OS and thereby invoke the actual processing of the packet (4). Using the pointer to the data of the packet (5), initial operations can be performed (6) before a routing decision is made. This includes for example integrity checks or the application of firewall rules. Now the actual processing takes place, a forwarding decision is made (7). A Longest Prefix Matching [17] algorithm takes the information of the IP header to find the best match in a routing table. After the next hop has been determined, the same operations as described in step (6) can be applied again (8). Furthermore, the layer 2 address for the next hop is being resolved. When the processing of the packet is finished, the egress network board, as determined through the routing lookup, is informed that the packet is ready for transmission (9). Similar to the processing of the ingress NIC, a descriptor to the memory region of the packet is stored in a queue (10), before it finally is transmitted (11).

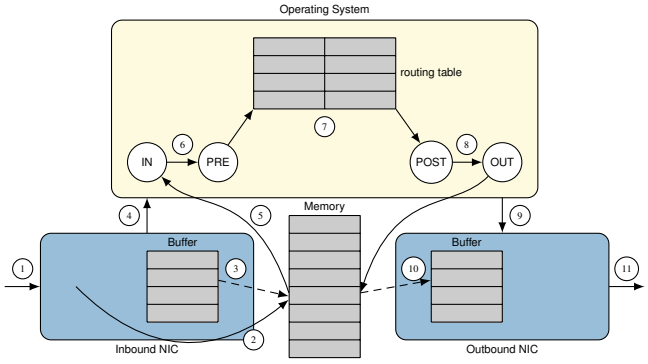


Fig. 1. Abstract model of the packet forwarding process

Our model assumes the forwarding process of network traffic consists of two main parts [18]. The NIC has to handle the transfer of the packets between itself and the main memory and the OS has to make the layer 2 and 3 forwarding decision. The abstract view reveals that the *access to main memory* and the *used data structures* are potential performance limiting factors [9], [16].

A. Packet Processing on Linux Operating Systems

Operating systems implement a general purpose network stack for processing and delivering of packets to user applications that comes with a subsystem for layer 3 forwarding. This allows Linux systems to act as software router.

The steady development process of the Linux networking code [19] requires steady updates to documentation and measurements. In addition to reading kernel source code, publications like [8], [17], [20] provide further insights and measurements. Furthermore, a book by Rosen [21] provides an in-depth explanation of the Linux networking kernel code. Our analysis is based on kernel version 3.7 [22]. For changes since this version the interested reader is referred to Märdian [19] who summarized new features that were added from version 3.7 to 3.16.

1) Reception API – NAPI: With the arrival of the packet, transfer of the packet to main memory, and invocation of further processing by informing the OS, have to be accomplished [8], [20], [21]. The Linux kernel uses the *sk_buff* structure for the internal representation of packets. The NIC stores descriptors, pointing to *sk_buff* structures, into buffers. It uses at least one buffer for packet reception (*rx_ring*) and one for transmission (*tx_ring*). When receiving a packet, the descriptor has to be initialized and allocated with an empty *sk_buff* (cf. Fig.1). The Direct Memory Access (DMA) engine of the NIC transfers the packet data to kernel memory space, the packet is stored in an *sk_buff*, and is ready for further processing. Otherwise it will be discarded by the NIC, because no packet descriptor was available [6]. This feature allows the NIC to drop packets without additional CPU load when the system is overloaded. Thus overload has no impact on the performance of the network stack [17].

To inform the kernel that a packet is available the NIC schedules a hardware interrupt through its interrupt generator. The CPU responds by calling the interrupt handler of the driver. Since the kernel version 2.4.20 the driver uses the New API (NAPI) [21]. Instead of directly enqueueing each packet into the backlog queue of the CPU, the interrupt handler invokes `napi_schedule` that adds a reference to the NIC to the `poll_list` of the CPU and triggers a soft interrupt. When this software interrupt gets served the CPU executes `net_rx_action`, which then successively uses the `poll` driver method of each device present in the `poll_list` in a round-robin fashion to poll a certain number of available packets from their ring buffer. For each of these packets `netif_receive_skb` is called, which then invokes the layer 3 handler of the network stack by executing `ip_rcv` [20].

2) Linux Network Stack: The Linux network stack is designed to support multiple networking protocols including the Address Resolution Protocol (ARP), the Internet Protocol, and Internet Control Message Protocol (ICMP), the user Datagram Protocol (UDP) and Transmission Control Protocol (TCP) [23]. The network stack processes each packet layer by layer. In order to forward a packet, however, only the layers up to the network layer are involved. For IPv6 packets separate functions exist that follow the same naming conventions (for instance `ipv6_rcv` instead of `ip_rcv`).

a) Prerouting Processing: The first function that gets invoked, `ip_rcv`, sanity checks for the length of the IP header, the checksum, and the length of the payload [17], [20]. If any of these checks fail, the packet gets dropped.

The packet next passes the netfilter subsystem. The purpose of this subsystem is to filter out packets and to perform changes [17]. It is used for firewalls implemented with iptables [24], network address translation (NAT) [25], modifying the contents of packet headers (mangling), connection tracking or gathering network statistics [20].

Afterwards, `ip_rcv_finish` continues and determines whether the packet has to be delivered locally or forwarded to another host. Hence, via a call to `ip_route_input_noref` the packet gets passed to the routing subsystem. The last action of `ip_rcv_finish` is to invoke `dst_input` of the *sk_buff*. Depending on the result of the routing lookup, this method calls the corresponding function for next steps.

b) Routing Subsystem: In Linux all threads and processes use the same shared routing table. Hence, the first task of `ip_route_input_noref` is to acquire a read-copy-update (RCU) lock [26]. As it synchronises the access with low overhead and wait-free reads, RCU-locks are widely used within the Linux kernel [27] and replace other locking mechanisms, whenever possible. Aside from sorting out for instance broadcast and multicast packets to handle them separately, a lookup in the routing table is initiated by calling `fib_lookup`. Linux uses the forwarding information base (FIB) trie [20]. The FIB trie contains the routing entries, each being a mapping of a subnet or IP address to a next-hop and outgoing interface. A lookup in the specified table is performed with `fib_table_lookup`. The basic idea of this function is to go through the FIB trie and try to find the best match for the destination IP address of the packet according to the longest prefix matching algorithm [20].

Depending on the result of the lookup, the *sk_buff* gets updated. The `dst_input` method is either set to `ip_local_deliver` for delivery to the local host or to `ip_forward` if the packet has to be forwarded. In the latter case the next-hop and outgoing interface get updated, too.

c) Postrouting Processing: If the packet gets delivered to the local host it gets passed to the next layers for further processing (for instance TCP or UDP), until it reaches the application in userspace via the socket API [28]. As this path is not subject of this paper we refer to [6], [17], [20] for further description.

In case of a forwarding decision to the next host the time to live (TTL) of the IP header is being decreased. If the TTL reaches zero, the packet gets dropped and an ICMP time exceeded message is triggered [20]. The length of the packet is compared to the maximum transmission unit of the outgoing link. Afterwards, potential IP options are processed and statistics get updated. Now a netfilter hook (`NF_INET_POST_ROUTING`) is invoked again. If fragmentation is needed and allowed it is handled via `ip_fragment`. To resolve the next-hop to a MAC address the packet is passed to the neighbouring subsystem via `__ipv4_neigh_lookup_noref`. If the entry is cached, the respective header field can be updated, otherwise the MAC address has to be resolved for instance using ARP for IPv4 or the neighbour discovery protocol for IPv6 packets. When the data link header has been completed, the packet is passed to the transmitting NIC by invoking `dev_queue_xmit`.

3) Transmission API: To control and schedule the traffic between kernel and NIC, queueing disciplines are used [29]. The default queueing discipline, *qdisc*, uses the FIFO strategy to manage the packets [20]. `dev_queue_xmit` enqueues each packet into the *qdisc* of the device that has been determined by the routing lookup. Then a spin-lock for the respective *qdisc* is acquired. These locks are the Linux implementation of active waiting-mutexes [30], hence, a transmission queue can only be used by one process at a time. When this lock is successfully acquired and the device is running and not stopped because for instance *tx_ring* is full, all the packets in the *qdisc* are handled. Another lock (`HARD_TX_LOCK`) has to be acquired before the `dev_hard_start_xmit` function of the driver is invoked. This method loads the packet descriptor into the `tx_ring` [20]. Finally, the NIC is informed of the packets that are ready for transmission.

The NIC informs the CPU with an interrupt when the packet transmission is completed After the interrupt, the meta data of the *sk_buff* structure is deallocated and the memory is freed.

B. Performance Limiting Factors

While the Linux design is convenient for running applications up to a rate of 1 Gbit/s, it rapidly reaches a limit when trying to work with rates up to 10 Gbit/s [9]. Furthermore, processing is not optimized for a forwarding-only task. Impediments for packet processing in software routers have been identified [1], [2], [5], [6], [9].

1) CPU: The most common bottleneck is the CPU. The more complex the processing of a packet is, the more CPU-cycles are consumed. A CPU operating at a frequency of CPU_{freq} Hz provides a number of CPU cycles per second $C_{available}/s$ and needs a total of C_{total} CPU cycles to process a single packet. The resulting maximum packet rate R_{max} can be calculated with

$$R_{max} = \frac{CPU_{freq}}{C_{total}} = \frac{C_{available}/s}{C_{total}} \tag{1}$$

This is valid when the CPU operates at its maximum frequency and full capacity. Thus, R_{max} can only be influenced by C_{total}.

2) Memory: The complete data of a packet, consisting of meta-data and payload, passes the system memory (cf. Section III-A). This leads to two different obstacles when forwarding packets at high-speed: costs for allocation and deallocation of memory that is influenced by the complexity of the data structure (*sk_buff*). Especially when working at high packet rates, constantly allocating and deallocating memory causes significant overhead that reduces the performance of the system [2]. The complexity of the *sk_buff*-structure that is compatible with numerous protocols and thus contains the meta-data of several protocols increases the workload. For the task of forwarding a packet, however, only the layer 2 and 3 headers are needed. Higher layer meta-data is not processed, as the packet is not passed up to these protocol handlers. Dorado et al. claim that "63% of the CPU usage in the reception process of a single 64B sized packet" [2] is consumed by *sk_buff* related operations. Another proof is the software switch, DPDK vSwitch [13] that uses a purged data structure to increase the speed compared to Open vSwitch.

3) Generality of the Software Design: At multiple occasions of the forwarding process data structures are protected with locking mechanisms to prevent faulty data caused by race conditions. One example, where multiple locks are used, is the access to the rx NIC [17]. For instance, the *qdisc* is protected with a spin-lock. Whenever a spin-lock is already locked, many CPU cycles are potentially wasted for active waiting of other processes [30]. Anyhow, locks can be omitted for dedicated software data planes where each CPU core gets packets in exclusively used rx buffers and stores them to dedicated tx buffers. So although Linux routers waste CPU cycles for obtaining and releasing these locks a Linux router scales linearly as there is no shared access on rx and tx rings.

4) Cache Size: With large data structures the size of the available CPU caches can limit the performance. If the data surpasses the size of the cache the number of cache misses can increase significantly and cycles are spent fetching the missed information. Cache thrashing causes the cycles per packet to increase as the actual processing of the packet is idling for data. This effect may occur due to the size of the routing table. For comparison today's backbone routers, which use the Border Gateway Protocol, hold up to 500,000 entries [31].

Additional processing tasks like firewalls, NAT, or collecting statistics, that can be hooked via netfilter rules in the Linux network stack at multiple occasions may require additional CPU cycles, (cache) memory, and locks.

IV. Performance Break Down

Profiling reveals how many CPU time has been spent on each function. Due to the high complexity of packet processing these results are hardly useful without further treatment. Filtering out all components that do not contribute to packet processing and grouping of the remaining functions allows for further analysis. The model which defines the groups is shown in Figure 2. It allows to compare different scenarios and to compare CPU cycles spent for each group of processing tasks.

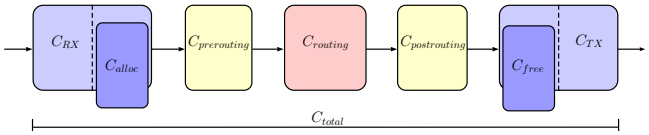


Fig. 2. Performance prediction model

We defined the following groups:

C_{RX} contains all functions for reception via the NIC and its driver. The tasks are considered to be finished after the driver handed the packet over to the OS for the actual processing of the packet headers.

C_{alloc} contains allocation and transfer of the packet data to the main memory. Though this could be part of C_{RX}, the separate group for buffer allocation accommodates for the significant impact of allocation on the overall performance.

$C_{prerouting}$ for processing steps after the packet was received from the driver but before the routing decision is made. The pre-routing processing includes for instance integrity checks and the traversal of firewalls or NAT.

$C_{routing}$ for lookup of the packet route – i.e. a FIB lookup.

$C_{postrouting}$ for tasks after the routing decision – for instance fragmentation or the resolution of MAC addresses.

C_{free} for packet buffer deallocation – analogue to C_{alloc}, the deallocation of the packet buffer.

C_{TX} transmitting side NIC tasks, analogous to C_{RX}.

C_{other} is a group for all functions that remained unassigned. It contains functions that are invoked in different groups and thus cannot be assigned to an own group and background tasks, which cannot be turned off. Fewer CPU time in this group leads to more precise statements and conclusions regarding the other groups.

Summing up the individual parts leads to the overall consumed cycles per packet

$$C_{total} = \sum_{i \in \{RX, alloc, \ldots, other\}} C_i, \tag{2}$$

With Equation 1 we can predict the maximum packet rate R_{max}.

Vice versa, one can measure R_{max} and either CPU_{load} or C_{active} - the amount of cycles that the CPU actively spent processing per second[1] - to calculate C_{total}:

$$C_{total} = \frac{C_{available}/s * CPU_{load}}{R_{max}} = \frac{C_{active}}{R_{max}} \tag{3}$$

A load that produces $CPU_{load} = 100\%$ avoids artefacts that can be caused by interrupts [8] or empty poll cycles [16]. To parametrize the model one has to measure how many CPU cycles are consumed by each function and then sort them into the groups of the model.

[1] C_{active} plus the amount of cycles spent idle must equal $C_{available}$

V. Parametrization of the Model

In this section we demonstrate the application of our model (cf. Section IV). The described (and further) measurements were performed by Dominik Scholz [32]. They were restricted to single core measurements as software routers scale linearly with the number of cores due to optimal parallelism [1]. The methodology was developed in previous measurements [1], [8], [9], [16], [33], [34].

A. Methodology

The measurements were performed on serves equipped with Supermicro X9SCM-F mainboards and Intel Xeon E3-1230 v2 CPUs with 3.3 GHz and 8 MB L3 cache, Dual channel 16 GB ECC DDR3 SDRAM clocked at 1333 MHz, and Intel X520-SR2 (DuT) and X520-SR1 (load generator and sink) based on the Intel 82599 Ethernet controller. Hyper-threading, turbo boost, and power saving features were disabled. To avoid fluctuations from CPU migrations, all processing tasks and interrupt handlers were always explicitly pinned to specific CPU cores. All offloading features of the NICs and pause frames were disabled. The size of the *rx_ring* was increased to its maximum of 4096 entries.

We ran Grml Live Linux 2013.02 with the Linux kernel version 3.7, and the ixgbe 3.14.5 NIC driver. Some experiments required a kernel with base pointers (compiler option `-fno-omit-framepointer`) to enable the profiling tool `perf` to create function call-stacks. Previous experiments showed that base pointers reduce the total throughput by about 15% [9].

All unnecessary background tasks were disabled to minimize possible interferences that affect the measurements. To generate packets up to the line rate of 10 GbE a customized version of the `pfsend` packet generator based on the pf_ring DNA packet processing framework has been used. To count the sent and received packets the statistics register of the NIC has been used.

The Linux profiling tool `perf` was used for all measurements. The sampling was restricted to be system-wide of all applications running on the single core to which the traffic was pinned. The result is the actual number of CPU cycles per second C_{active}. This value is used to calculate the cycles per packet C_{total} using Equation 3 and to scale the values obtained with `perf record`[2]. When we used the alternative kernel version, compiled with base pointers, the resulting binary file of `perf record` contained information about the function call-stack.

We used profiling data to create flame graphs. Although Flame graph visualizations are too detailed for usage in papers the visualizations of the call stacks were helpful to identify CPU-consuming functions. Such an exemplary flame graph is displayed in Figure 3. We also used a modified colouring schema for Flame Graphs that clusters functions of the Flame Graphs and paints them using identical colour.

[2] `perf record` is a tool, part of `perf` that allows sampling of counters with configurable rate and periods

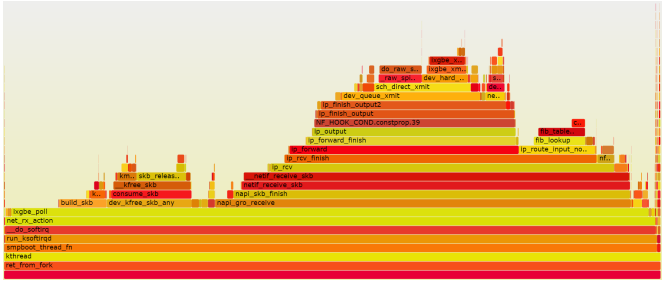


Fig. 3. Flame graphs are useful to get an overview to CPU cycle consumption: this exemplary flame graph visualizes the distribution of CPU cycles spent on the different functions of the Linux kernel

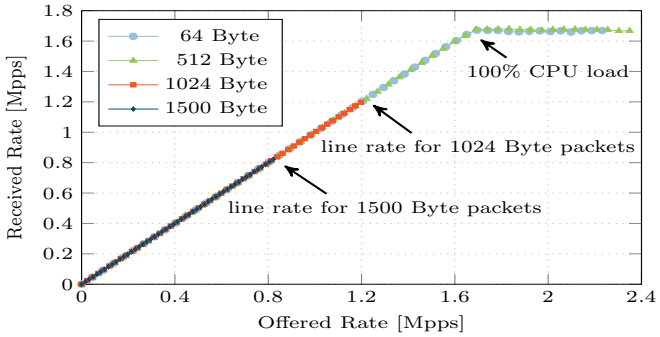


Fig. 4. Packet rates for different packet sizes

All profiling tests were run five minutes per offered rate to obtain reliable values. Additional sampling showed that under full workload the relative cycles of the functions were equal with a deviation of less than $\pm 1\,\%$ in a 95 % confidence interval. Therefore, the measurements with the maximum received rate shown in Figure 4 - thus maximum CPU load - were used to parametrize the model. For 64 Byte and 512 Byte packets the line rate of 10 Gbit/s was not reached. However, the router was at full load and, when increasing the offered load even more, started dropping packets. These packets are dropped by the NIC without stressing the CPU. Hence, for 64 Byte and 512 Byte packets the model is parametrized with mean values.

B. Increasing the Packet Size

Our first investigation analyses the system behaviour different packet sizes, ranging from the minimum size of 64 Bytes to a close-to-maximum size of 1500 Bytes. Previous work showed that for small packet sizes the line rate of 10 Gbit/s can not be reached [1]. Instead, when the CPU is completely utilized, additional packets are dropped by the NIC. For larger packets, however, the line rate can be reached, leading to a not fully utilized CPU.

A closer look at the different parts of our performance prediction model confirms that packet size has no influence on the packet rate (cf. Section II): The rx NIC handles all incoming packets the same, while *sk_buff*'s are always allocated to fit a maximum sized packet. The network stack only needs the information of the layer 2 and 3 headers to make a forwarding decision.

1) Impact on the Packet Rate: Figure 4 shows the received packet rates for the tested packet sizes. The offered rate equals the received rate of packets until the line rate is hit or the CPU is at full capacity. As expected, for the smaller packet sizes of 64 and 512 Bytes, the line rate is not reached. Instead, the CPU reaches full capacity when offering 1.67 Mpps. The received rate remains at this maximum when offering even more packets per second as additional packets get dropped by the rx NIC. For 1024 and 1500 Byte packets the line rate is reached at 1.2 Mpps and 0.82 Mpps respectively but the CPU is not fully utilized for these packets.

TABLE I
MAXIMUM PACKET RATE, MAXIMUM CPU LOAD AND CYCLES PER PACKET FOR DIFFERENT PACKET SIZES

Packet Size	R_{max} [Mpps]	CPU_{load}	C_{total}
64 Byte	~ 1.67	$\sim 100\%$	1979
512 Byte	~ 1.67	$\sim 100\%$	1969
1024 Byte	~ 1.20	$\sim 75\%$	2056
1500 Byte	~ 0.82	$\sim 55\%$	2203

Table I summarizes the maximum packet rate and the respective CPU load for all tested packet sizes. The resulting cycles per packet were calculated with Equation 3 with $C_{available} = 3.3 * 10^9$ CPU cycles per second for all further calculations. C_{total} is constant for different packet sizes, except in scenarios where the packets per interrupt decrease and thus the costs for interrupts per packet increase. Explicit listing of interrupt costs can help to explain this effect, but is not applicable for polling based packet reception (cf. Section II).

2) Distribution of CPU-cycles: Figure 5 shows the results of the `perf record` measurements for 64 Byte IPv4 packets. Until 1.67 Mpps, the consumed CPU cycles increase linearly for almost all functions with the offered packet rate. The functions `ip_rcv`, `fib_table_lookup` or `dev_queue_xmit` account for large shares in CPU utilization. The function with the highest percentage is

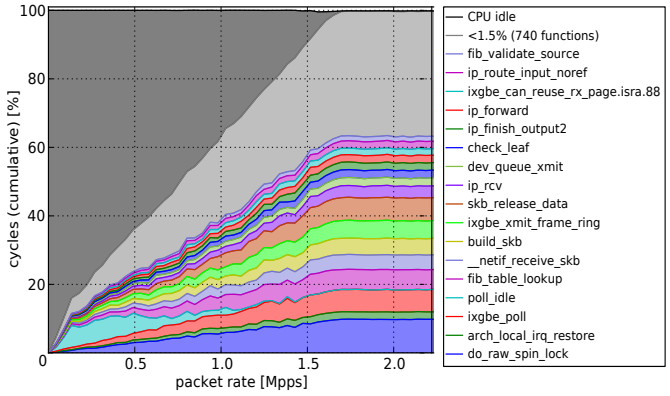


Fig. 5. Distribution of CPU-cycles across functions for 64 B packets

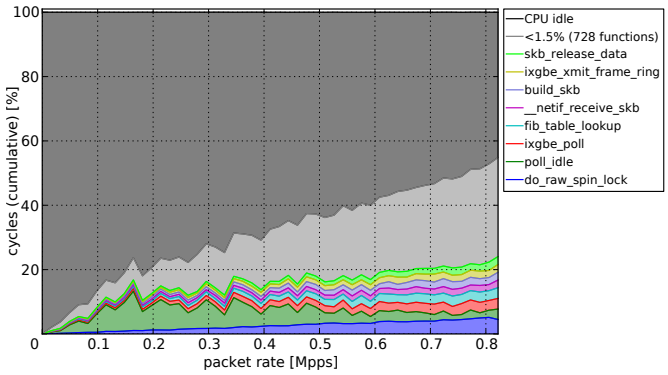


Fig. 6. Distribution of CPU-cycles across functions for 1500 B packets

`do_raw_spin_lock`, which is implementing active waiting to acquire a lock for a resource that is shared among multiple processes. This shows that locking mechanisms as they exist in general purpose systems, like Linux, are limiting the performance of the network stack significantly. As modern NICs offer multiple rx_rings these locks could be removed from the general packet transmission process in Linux as long as it is ensured that each rx_ring is used *exclusively* by one core.

The number of polls at a certain frequency is adjusted dynamically by the NAPI driver [8]. This way, the latency is reduced when only processing few packets, while under higher loads the overhead is reduced. This effect of the interrupt throttle rate explains the rapidly increasing CPU activity for the first ∼0.2 Mpps [9]. With an increasing packet rate the number of interrupts increase, too. When reaching a certain maximum, the number of packets polled per interrupt is increased instead. This then leads to the less steep slope of CPU activity after 0.2 Mpps.

As `perf` samples all the functions of the kernel, more than 750 different names appear in the output, many of which consume close to no cycles. The values of all functions are constant after hitting 1.67 Mpps. This can be explained with the *rx_ring* of the NIC. If not enough packet descriptors are polled from the ring, no free descriptors are available for incoming packets, hence, these are dropped. This is done without any time loss, resulting in no impact on the overall performance. Therefore, measurements with packet rates that lead to full CPU load are treated as if they were multiple tests for the maximum received packet rate. These are then used to calculate average values with errors bars showing the 95 % confidence intervals.

The effect that 1500 Byte packets need slightly more cycles compared to smaller packets can be explained with the associated profiling graph in Figure 6. As the line rate is hit at a rate of 0.82 Mpps, the CPU is not under full load and the graph equals the one for 64 Byte packets for 0 Mpps to 0.82 Mpps (cf. Figure 5). Therefore, the idle functions, in particular `poll_idle`, are still present and consume additional cycles compared to smaller packet sizes.

The last step to parametrize the performance prediction

TABLE III
TOP 5 CYCLE-CONSUMING FUNCTIONS IN C_{other} (64 B PACKETS)

Function	Cycles	appears in
do_raw_spin_lock	194	all Groups
arch_local_irq_restore	43	C_{RX}, $C_{prerouting}$
arch_local_irq_save	21	C_{alloc}, C_{free}
__phys_addr	20	C_{RX}, C_{free}, C_{TX}
spin_unlock	15	C_{alloc}, $C_{postrouting}$, C_{free}, C_{TX}

model is to divide each function into a group of the model. Table II shows the resulting values for the model when using the measurements with different packet sizes. Figure 7 shows a plot of these values.

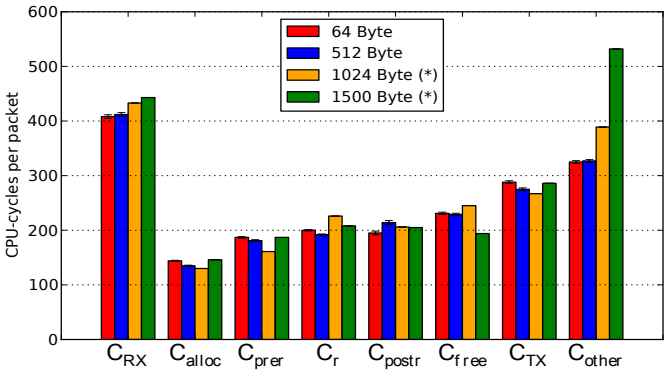


Fig. 7. Per packet CPU-cycles distribution for different packet sizes

The processing by the rx and tx NICs and their respective drivers consumes approximately 35 % of the whole cycles with minimal increases for larger packets at the receiving side, which are explained by the increased DMA transfer time. The allocation and deallocation is independent from the size of the packet, because a maximum sized *sk_buff* is allocated anyway. However, both together consume up to 20 % of the total cycles, showing that per packet allocation and deallocation are a costly task. The processing of the network stack, represented by $C_{prerouting}$, $C_{routing}$ and $C_{postrouting}$, uses only the information of the necessary headers, hence the size of the payload does not matter. Each individual part takes about 10 % of the cycles, without the use of any additional processing.

A downside of our model is the inaccuracy caused by the functions in C_{other}. To solve this issue, a more sophisticated method to distribute the functions into the groups of the model can be used, factoring in what share of each function is used per group.

C_{other} represents the final 15 % of CPU cycles, including functions that appear in more than just one group or only consume a close to zero percentage. The five functions with the biggest share are shown in table III. Furthermore, it is shown in which parts of the model they occur. Function `do_raw_spin_lock` consumes close to 10 % of the CPU cycles. Approximately 90 % of the calls to `do_raw_spin_lock` occur when transmitting a packet by the tx NIC, hence 90 % of the cycles for this function

TABLE II
PER PACKET CPU-CYCLES DISTRIBUTION FOR DIFFERENT PACKET SIZES (* AND RATES)

Packet Size	C_{RX}	C_{alloc}	$C_{prerouting}$	$C_{routing}$	$C_{postrouting}$	C_{free}	C_{TX}	C_{other}	C_{total}
64 Byte	408	144	187	200	195	231	288	325	1979
512 Byte	412	135	181	192	214	229	275	327	1969
1024 Byte (*)	433	130	161	226	206	245	267	389	2056
1500 Byte (*)	443	146	187	208	205	194	286	532	2203

can be added to C_{TX}. Doing so leads to 40-45 % for rx and tx NIC processing. This conforms to previous research [2], [10]. Anyway, for higher accuracy we continue to count `do_raw_spin_lock` to C_{other}.

For larger packet sizes, in particular 1500 Byte, C_{other} consumes even more cycles. This is again explained by the idle functions, for instance `poll_idle` with $\sim$ 40 cycles, are added to this group. This was done mainly for the reason that these functions do not actually process the packet and would just obscure the results, making them less comparable with the smaller packet sizes.

In summary we see a constant behaviour across all parts of the model. Only C_{other} increases due to the increased relative impact on a packet rate that is decreased due to the limiting Ethernet link bandwidth (see (*) in table II). Therefore, our following tests only use 64 Byte packets to reduce the overall number of test cases. Our tests show impediments of locking mechanisms and per-packet allocation and deallocation of packets. This is the reason why most high-speed packet processing frameworks implement different techniques to solve especially these issues [10].

C. Increasing the Number of Flows

In the following we increase the number of flows, each represented by a packet with an unique destination address, to analyse the behaviour of the routing subsystem for a growing routing table. The measurements with 64 Byte packets of the previous section are used as reference, representing one flow. The general assumption is, that only the behaviour of the routing subsystem differs for varying numbers of flows. As all packets are pinned to the same core, neither the NIC nor the network stack treats the packets differently in any way. The traffic itself uses repetitive, linear increasing IPv4 addresses as destination. This was done to test the worst case, constantly changing destinations, which means that the results of the lookup can not be cached. However, as kernel version 3.7 uses no routing cache anyway, this behaviour is given by the design of the routing subsystem. The expected result is, that as long as the necessary data of the FIB trie fits into the cache of the CPU, the performance does not decrease significantly.

The packet rates for different numbers of flows are shown in Figure 8. Concurring with previous results, the rate decreases slightly ($\sim$ 0.1 Mpps) for tests with more flows. The run with more than one million flows however shows a maximum packet rate of only 1.39 Mpps. The packet rate decreases when increasing the offered rate even further. This effect can be explained by the cache behaviour of the CPU (cf. V-D).

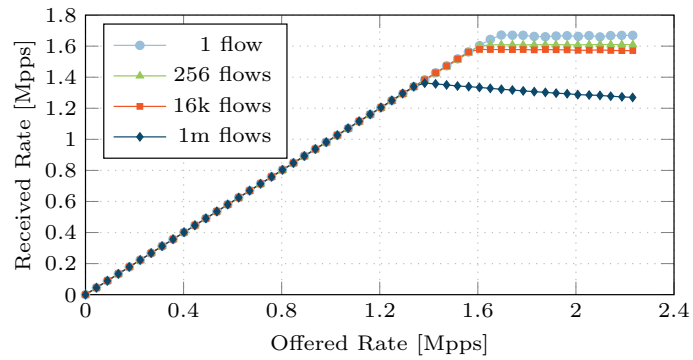


Fig. 8. Packet rates for different number of flows

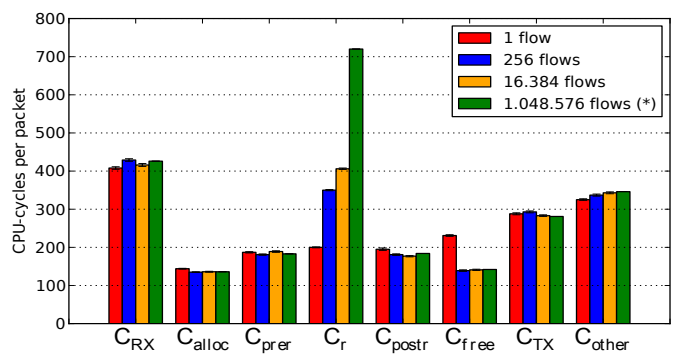


Fig. 9. Per packet CPU-cycles distribution, different number of flows, 64 B

Parametrizing the model with the results of these measurements lead to the values shown in table IV and presented in Figure 9. As expected, more flows have no impact on the processing by the NIC and its driver. The only anomaly is given by C_{free}. When having a closer look at which functions, that are divided into C_{free}, cause this behaviour, only `skb_release_data` shows a significant deviation. In the last row of table V, for 1 flow this function consumes approximately 133 cycles, while for the other tests this is reduced to under 40 cycles. A possible explanation for this is the dynamic memory management of the OS. For one flow, the packet rate is slightly increased, hence, more packets per second have to be managed at the same time. This means that more locks must be acquired to release the data again, which could cause this increased consumption of cycles.

The pre- and postrouting processing is constant, as the same operations are performed, independently of the destination address. Only the routing subsystem shows an increased

TABLE IV
PER PACKET CPU-CYCLES DISTRIBUTION FOR DIFFERENT NUMBER OF FLOWS, 64 BYTE

Number of Flows	C_{RX}	C_{alloc}	$C_{prerouting}$	$C_{routing}$	$C_{postrouting}$	C_{free}	C_{TX}	C_{other}	C_{total}
1	408	144	187	200	195	231	288	325	1979
256	429	135	181	350	181	139	293	337	2048
16.384	416	136	189	406	177	141	283	343	2093
1.048.576 (*)	426	136	183	720	184	142	281	346	2420

TABLE V
FUNCTIONS WITH THE LARGEST DIFFERENCES IN THEIR RESPECTIVE GROUP FOR DIFFERENT NUMBERS OF FLOWS

Function	Group	1 flow	256 flows	1.048.576 flows
fib_table_lookup	$C_{routing}$	116	238	423
check_leaf	$C_{routing}$	45	89	361
skb_release_data	C_{free}	133	37	35

consumption of cycles for increasing amounts of routing table entries. This is matching the assumptions, as no routing cache is implemented. Only two functions, listed in table V have a significant impact on this increase of up to 3.5 times the value for one flow. Both are directly involved for finding the best match in the FIB. This means that finding the best match in the FIB trie lasts longer, as the size of the tree-like structure that represents the routing table is increased, hence, it takes more time to traverse it until the correct leaf is found.

D. Limitations through the Cache Size

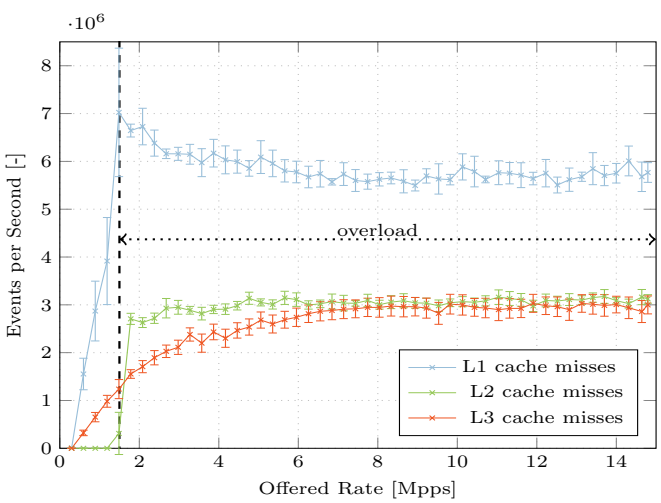


Fig. 10. Cache misses for 1.048.576 Flows.

The effect that for 1.048.567 flows the received rate decreases after hitting the peak rate when further increasing the offered rate (cf. Figure 8), is deviant from all other measurements. To better understand this behaviour we repeated the test with a bit rate up to the maximum of the 10 GbE links. The received rate decreases until it approaches approximately 1.1 Mpps at an offered rate of 8 Mpps and stays at this level. A decreased packet rate means the cycles per packet increase. When having a look at which functions consume these additional cycles, only functions in $C_{routing}$ show an increased consumption.

Figure 10 shows the misses across the level 1, 2 and 3 caches of the CPU. The curve of the L1 misses is similar to the packet rate, while the L2 cache misses are at a nearly constant level of half of the L1 misses. The most interesting is the progression of the last level cache misses. In the beginning they are rapidly increasing which is not changing when surpassing the mark of the maximum packet rate - in this case the maximum L1 cache misses. Instead the L3 cache misses approach the L2 misses, until all L2 cache misses miss the L3 cache, too. We attribute this effect to the increased number of packets that are handled by the router, resulting in more requests per second to the routing table. Anyhow, the structure is too large to fit completely into the L3 cache. Already a low estimated value for one routing entry of 20 Bytes (4 Byte IPv4 address, 4 Byte subnet mask, outgoing interface, metric, metadata, ...) leads to a routing table of approximately 21 MiB for 1.048.576 flows, which is vastly larger than the 8 MiB level 3 cache[3]. Together with the fact, that always different destination addresses are looked up, referencing different parts of the FIB trie, this causes more and more L3 cache misses. This in turn slows down the whole processing of the packets as they are backed-up, causing the cycles per packet to increase. The worst performance is reached when all L3 references miss the cache and thereby cause the maximum number of memory accesses.

In summary the performance is not decreased significantly when increasing the number of flows. The only identified difference is, as expected, finding the best match in the routing table, as this structure keeps growing with the number of flows. Once this number exceeds a certain value, however, another limiting factor, the size of the L3 cache, gains importance. If the FIB is larger than the cache, the overall performance is reduced because of worse cache behaviour. Generally this shows, that the size of used data structures must be kept in mind, as bad cache behaviour may lead to unexpected effects which drastically reduce the performance.

While a routing table with more than one million entries may be unrealistic this loss of performance might already occur for a number of flows between 65.536 and 262.144. Hence, this effect is relevant, as BGP routers can have more than 500.000 routing entries nowadays [31]. These values are continuously rising, causing other problems, too, as for instance backbone routers run out of memory [35].

[3]For reference, using the same estimated size for a routing entry, 16.384 entries need a total of only 328 KiB, hence the complete routing table fits into the level 3 cache.

VI. Summary

We presented an analytical model for software based packet processing systems that can predict upper bounds for the maximum throughput. Our model can be parametrized to fit arbitrary systems. Our case study where we broke down the performance of a Linux router described the influence and relationship of FIB size, network stack, and packet processing tasks on the performance. We encourage to use our profiling data for model parametrization.

Acknowledgments

This research has been supported by the DFG as part of the MEMPHIS project (CA 595/5-2) and the BMBF in context of the EUREKA-Project SASER (01BP12300A). We acknowledge the valuable contributions from Alexander Beifuß, Torsten Runge, and Sebastian Gallenmüller.

References

[1] T. Meyer, F. Wohlfart, D. Raumer, B. E. Wolfinger, and G. Carle, "Measurement and Simulation of High-Performance Packet Processing in Software Routers," *Proceedings of Leistungs-, Zuverlässigkeits- und Verlässlichkeitsbewertung von Kommunikationsnetzen und Verteilten Systemen*, September 2013, 7. GI/ITG-Workshop MMBnet 2013.

[2] J. L. Garcia-Dorado, F. Mata, J. Ramos, P. M. S. del RÃo, V. Moreno, and J. Aracil, "High-performance network traffic processing systems using commodity hardware," *Data Traffic Monitoring and Analysis*, pp. 3–27, 2013.

[3] IEEE, "IEEE P802.3ba 40Gb/s and 100Gb/s Ethernet Task Forceofficial web site," http://www.ieee802.org/3/ba/, accessed: May 2015.

[4] M. Dobrescu, N. Egi, K. Argyraki, B. G. Chun, K. Fall, G. Iannaccone, A. Knies, M. Manesh, and S. Ratnasamy, "RouteBricks: Exploiting Parallelism To Scale Software Routers," *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*, pp. 15–28, 2009.

[5] R. Bolla and R. Bruschi, "Linux Software Router: Data Plane Optimization and Performance Evaluation," *Journal of Networks*, vol. 2, no. 3, 2007.

[6] W. Wu and M. Crawford, "The Performance Analysis of Linux Networking - Packet Receiving," *International Journal of Computer Communications*, 2006.

[7] R. Bolla and R. Bruschi, "PC-based Software Routers: High Performance and Application Service Support," *Proceedings of the ACM workshop on Programmable routers for extensible services of tomorrow*, pp. 27–32, 2008.

[8] A. Beifuß, D. Raumer, P. Emmerich, T. M. Runge, F. Wohlfart, B. E. Wolfinger, and G. Carle, "A Study of Networking Software Induced Latency," in *2nd International Conference on Networked Systems (NetSys)*, March 2015.

[9] Paul Emmerich and Daniel Raumer and Florian Wohlfart and Georg Carle, "Assessing Soft- and Hardware Bottlenecks in PC-based Packet Forwarding Systems," in *Fourteenth International Conference on Networks (ICN 2015)*, Barcelona, Spain, Apr. 2015.

[10] L. Rizzo, "Revisiting Network I/O APIs: The Netmap Framework," *Commun. ACM*, vol. 55, no. 3, pp. 45–51, Mar. 2012.

[11] "Impressive Packet Processing Performance Enables Greater Workload Consolidation," in *Intel Solution Brief*. Intel Corporation, 2013, Whitepaper.

[12] "ntop.org: PF_RING ZC," http://www.ntop.org/products/pf_ring/pf_ring-zc-zero-copy/, last visited 2015-07-10.

[13] I. O. S. T. Center, "Packet Processing - Intel DPDK vSwitch," https://01.org/packet-processing/intel%C2%AE-onp-servers, accessed: September 2014.

[14] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek, "The Click modular router," *ACM Transactions on Computer Systems (TOCS)*, vol. 18, no. 3, pp. 263–297, 2000.

[15] L. Rizzo and G. Lettieri, "VALE, a switched ethernet for virtual machines," *CoNEXT*, pp. 61–72, 2012.

[16] S. Gallenmüller, P. Emmerich, F. Wohlfart, D. Raumer, and G. Carle, "Comparison of Frameworks for High-Performance Packet IO," in *ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS 2015)*, Oakland, USA, May 2015.

[17] C. Benvenuti, *Understanding Linux Network Internals*. O'Reilly Media, Inc., 2005, vol. 1.

[18] R. Bolla and R. Bruschi, "An Effective Forwarding Architecture for SMP Linux Routers," *Telecommunication Networking Workshop on QoS in Multiservice IP Networks*, pp. 210–216, 2008.

[19] L. Märdian, P. Emmerich, and D. Raumer, "What's New in the Linux Network Stack?" in *Proceedings of the Seminars Future Internet (FI) and Innovative Internet Technologies and Mobile Communications (IITM), Summer Semester 2014*, München, Germany, Apr. 2015.

[20] M. Rio, M. Goutelle, T. Kelly, R. Hughes-Jones, J. Martin-Flatin, and Y. Li, "A Map of the Networking Code in Linux Kernel 2.4.20," *Technical Report DataTAG-2004-1*, March 2004.

[21] R. Rosen, *Linux Kernel Networking: Implementation and Theory*. Apress, 2013, vol. 1.

[22] F. Electrons, "Linux Cross Reference Version 3.7," http://lxr.free-electrons.com/source/?v=3.7, accessed: August 2014.

[23] J. W. Buse, "Linux Network Stack," http://www.linux.org/threads/linux-network-stack.4620/, accessed: May 2015.

[24] I. set, "iptables," http://ipset.netfilter.org/iptables.man.html, accessed: May 2015.

[25] K. Egevang and P. Francis, "RFC 1631: The IP Network Address Translator (NAT)," 1994.

[26] LWN.net, "Using read-copy-update," http://lwn.net/Articles/37889/, accessed: May 2015.

[27] P. E. McKenney, "RCU Linux Usage," http://www.rdrop.com/users/paulmck/RCU/linuxusage.html, accessed: May 2015.

[28] die.net, "socket - Linux man page," http://linux.die.net/man/7/socket, accessed: May 2015.

[29] T. L. D. Project, "Components of Linux Traffic Control," http://www.tldp.org/HOWTO/Traffic-Control-HOWTO/components.html, accessed: May 2015.

[30] Wikipedia, "Spinlock," http://goo.gl/f1Dqhx, accessed: May 2015.

[31] APNIC, "BGP Routing Growth in 2011," http://labs.apnic.net/blabs/?p=25, accessed: May 2015.

[32] D. Scholz, "A Model for Performance Prediction in PC-based Packet Processing Systems," in *Bachelor's Thesis*, München, Germany, 2015.

[33] P. Emmerich, D. Raumer, F. Wohlfart, and G. Carle, "A Study of Network Stack Latency for Game Servers," in *13th Annual Workshop on Network and Systems Support for Games (NetGames'14)*, Nagoya, Japan, Dec. 2014.

[34] P. Emmerich, D. Raumer, F. Wohlfart, and G. Carle, "Performance Characteristics of Virtual Switching," in *2014 IEEE 3rd International Conference on Cloud Networking (CloudNet'14)*, Luxembourg, Oct. 2014.

[35] BGPMON, "What caused today's Internet hiccup," http://www.bgpmon.net/what-caused-todays-internet-hiccup/, accessed: August 2014.

How Do Multiple Network Cards Influence the Software Router Performance?

Torsten M. Runge[1], Florian Wohlfart[2], Daniel Raumer[2], Bernd E. Wolfinger[1], and Georg Carle[2]

[1]Universität Hamburg, Department of Computer Science, Telecommunications and Computer Networks
{runge|wolfinger}@informatik.uni-hamburg.de
[2]Technische Universität München, Department of Computer Science, Network Architectures and Services
{wohlfart|raumer|carle}@net.in.tum.de

***Abstract*—The usage of cost-efficient and dynamically adaptable commodity PC hardware instead of specialized networking hardware has aroused strong interest in software routers. Testbed measurements as well as modeling and simulations of such system are important methodologies for finding bottlenecks in the packet processing to predict and optimize the performance. In this paper, we investigate the impact of the number of network cards with respect to the software router performance. We predict the performance of software routers based on a validated model which we obtain from real testbed measurements.**

***Keywords*— commodity hardware; packet processing; resource contention; network card; simulation model; ns-3; Linux NAPI**

I. Introduction

Commodity PC hardware has received several performance improvements. For instance, the trend towards many CPU cores as well as the multi-queue support of network interface cards (NIC) for parallel packet processing on multi-core architectures make off-the-shelf PC hardware interesting for the application as servers or routers. Furthermore, widely used open source operating systems such as Linux provide basic packet processing functionalities in software which can transform PC hardware into a so-called software router. The advantages of software routers are that they are more cost-efficient and more flexible to extend with new functionalities in comparison with special networking hardware such as hardware routers. However, hardware routers are often capable of higher packet processing performance and lower energy consumption. Nonetheless, in small to mid-range networks, such as home or campus networks, software routers represent an attractive alternative to expensive hardware routers.

The challenge for software routers is to process packet rates resp. data rates of million packets per seconds (Mpps) resp. multiple gigabits per second (Gbps) with no packet loss and acceptable packet latency. Therefore, the system internal components such as bus systems (e.g. PCIe, QPI) must provide the required data rates between the hardware components. Depending on the type of packet processing (e.g. routing, firewall, IPsec encryption) the software router must do several complex treatments per packet. In particular, the scaling of the number of NICs of software routers is important to compete with hardware routers.

For the optimization of the software router performance it is necessary to understand the packet processing steps in PC systems such as the Linux networking kernel as part of the operating system (OS) in detail. However, a software router is a complex system which consists of diverse hardware and software components interacting and influencing each other. Therefore, modeling and simulation may help to understand the complex interplay during the packet processing. Based on a sufficiently fine-grained model a complex system can be analyzed to find performance limiting factors and optimize the system accordingly to eliminate such bottlenecks. Finally, the model can be used to predict the performance behavior with respect to diverse scenarios which cannot be investigated in real testbeds.

In this paper, we measure and simulate the performance of software routers with respect to multiple network interface cards. Firstly, we conduct real testbed measurements with a Linux software router. Besides, we create a model for a software router with multiple network cards. This model is derived from our general concept for modeling of resource contention in resource-constrained nodes which is implemented as a resource management module for the widely used network simulator ns-3. The software router model is calibrated and validated with real testbed measurements. Based on that, we evaluate and predict the impact of multiple network cards on the performance of software router architectures.

The rest of the paper is structured as follows. Section II gives an overview of the state of the art in modeling, measuring, and implementation of software routers. Section III describes the procedures of the packet processing inside a Linux software router. In Section IV we describe our testbed which we used for the calibration of our software router model. Section V introduces and applies our general modeling approach to model a Linux software router with multiple network cards. Section VI presents a case study to illustrate the influences of multiple networks on the software router performance. The case study includes the calibration and validation of our simulation model based on the real testbed measurements. Finally, we conclude the paper and give an outlook to future work in Section VII.

II. Related Work

Many researches have shown that software routers are able to reach the performance of specialized networking hardware [1]–[3]. The RouteBricks project [1], [3] investigated the scaling of software routers based on commodity PC hardware. They applied modern NICs with multi-queueing support and exploit the parallel packet processing capabilities across multiple cores within a server as well as a software router cluster consisting of multiple servers. Bolla and Bruschi [4] identified the CPU to be the performance bottleneck in packet processing on commodity PC hardware which was also observed by RouteBricks [1] and us [5]. Therefore, small packets dominate the packet forwarding performance in software routers because a constant number of CPU cycles is needed per packet for the processing from the incoming port to the output port. Han et al. [2] investigated several improvements of the packet processing software by reducing the required number of CPU cycles per packet. Salim et al. [6], [7] investigated the Linux NAPI with respect to the reduction of the number of interrupts in high load situations. Salah et al. [8] evaluated and compared the performance of IP-packet forwarding of a Linux host equipped with three NICs.

Chertov et al. [9] proposed a router model which is able to predict the forwarding performance of arbitrary router devices based on calibration parameters retrieved from black box measurements. Salah et al. [10], [11] developed validated analytical models to estimate the performance of software based network hosts based on constant bit rate (CBR) and Poisson traffic. These models consider interrupt moderation mechanism of Linux and BSD with respect to the interrupt rate, the interrupt service routine (ISR) as well as packet processing times. We proposed a general and broadly applicable modeling approach [12] to investigate the node-internal resource contention during the packet processing. We implemented the modeling approach as a resource management extension module for the widely used network simulator ns-3. Based on that we evaluated the influence of the Linux NAPI with respect to packet latency [13].

Our best practice for measurements that we developed and proved in previous publications [5], [13]–[15] was influenced by the work from Bolla and Bruschi [16] and Dobrescu et al. [1], [17]. Both papers present detailed know how on performance measurements of software routers. For load generation we rely on *zsend*, which is based on the high speed network IO framework *PF_RING ZC* [18], which claims to produce CBR traffic. As zsend is a software-based load generator it cannot guarantee that the generated frames are accurately spaced. This is a general problem inherent to all software load generators [19]. We are actively working on an improved load generator to mitigate some of these issues by making use of specialized NIC hardware features (e.g. rate-limiting to avoid micro bursts) [20].

While we have studied the scalability of software packet processing based on the number of cores [5], the number of virtual machines [15], or by reducing the processing overhead [21] in the past, we are not aware of a detailed study of the performance overhead introduced by scaling the number of NICs on a packet processing system.

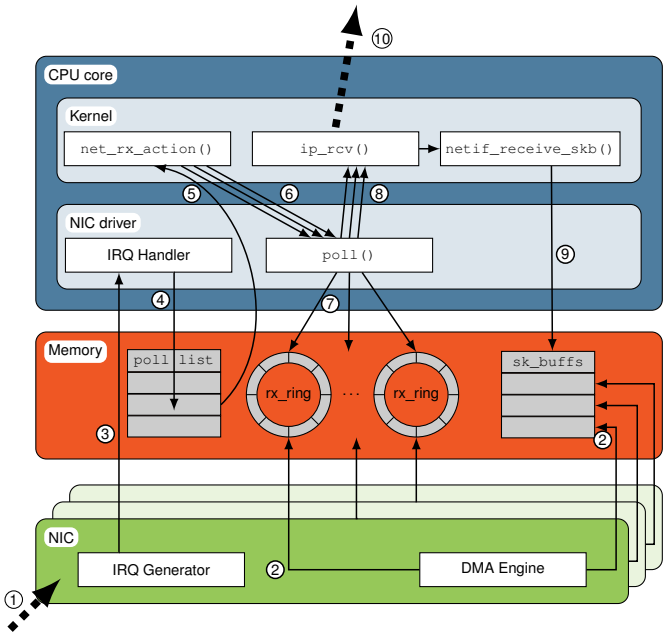


Fig. 1. Functional visualization of the processing path in a Linux router

III. Packet Processing in a Linux Software Router

As a first step to the analysis of Linux router scaling with the number of NICs we analyze the internals of packet processing. We sketch a simplified transmission path through the Linux router, which highlights aspects of scalability. Relevant elements are visualized in Fig. 1. For a more detailed description of packet processing functions in the Linux kernel we refer to our previous work [13], [22] in which we modeled the reception process in Linux as it is defined by the NAPI.

A. Packet Reception Path

Packets arriving at the NIC port (Fig. 1,①) are transferred to the memory via Direct Memory Access (DMA). The memory now contains the packet data and meta information which is stored in the `sk_buff`. A packet descriptor is an entry in the corresponding `rx_ring` which points to the respective `sk_buff`; ②. The NIC now triggers a hardware interrupt to the assigned CPU core; ③. The interrupt handler takes over and adds an entry to the `poll_list` that states that packets in the respective ring need to be processed; ④. The soft IRQ scheduler processes the soft IRQs which are processed by the `net_rx_action()`; ⑤.

The `net_rx_action()` function processes the entries of the `poll_list`. Batches of packets with a defined maximum batch size each get polled from the rings; ⑥. The actual poll happens via the NAPI function `poll()` that is defined as part of the NIC driver; ⑦. During that polling interrupts

are deactivated. So higher loads result in a more polling-based packet processing while high interarrival times result in a mainly interrupt-driven packet reception. This essential function of the NAPI provides a flexible tradeoff between throughput and latency: at high load the NAPI avoids interrupts and aims for the maximum throughput, at low load the NAPI generates more interrupts to decrease latency. In order to avoid a CPU core that is locked by a never ending polling which would steal the CPU core capacity from other starving processes (e.g. user space programs that actually process the received packets) polling is stopped after a certain `budget` was used.

B. From Reception to the Transmission

In case of IP packets `ip_rcv()` is called for each packet; ⑧. An IPv6 packet would take a similar path (via `ipv6_rcv()`). For each packet in the polled batches of packets (stored in an `sk_buff`) the function `netif_receive_skb()` is invoked and MAC level information (e.g. VLANs) are processed; ⑨. The next steps can be categorized into prerouting, routing, and postrouting; ⑩. Packets can leave the routing path: e.g. if they are addressed to the host they get filtered out in the routing step and are directed on a path for local delivery. As all further processing of each single packet is independent from the NIC that received the packet, we do not discuss it in more detail here. The interested reader is referred to [23].

C. Transmission Path

After the routing decision, the packet gets prepared for transmission: A pointer to the `sk_buff` is placed in a `tx_ring` of the egress NIC port via the `dev_queue_xmit()` (ring selection) and the `hard_start_xmit()` (actual writing) functions. Locking of the ring only occurs if a `tx_ring` is not exclusively assigned to a CPU core. The NIC sends the packets of the Tx ring and triggers an IRQ after the successful transmission. Then the memory used for the sent packets must be freed from the Tx ring to store new packets for transmission. The Tx work limit defines the number of Tx descriptors which are cleaned in a single batch.

D. Uniting Rx Flows from Different NICs

Modern NICs come with support for multiple Rx and Tx rings that help to avoid any overhead for locking of the rings due to parallel processing. From the CPU's point of view the splitting and merging of parallel tasks is offloaded to the NICs. Thereby, the number of theoretically available `rx_rings` and `tx_rings` is some orders higher than the numbers of cores. Dedicating each CPU core to the traffic arriving at a certain NIC is inefficient when the load is not equally distributed among all NICs. For instance, one core may already be fully utilized while other cores are idle because traffic volumes in their NICs are low. Therefore, it is advisable to evenly distribute the incoming traffic to all CPU cores.

In most scenarios a core may have to serve packets from at least so many rings as there are NIC ports in the software router. Therefore, different packet paths have to be united to pass a single CPU core. This happens the same way packet paths coming in via different rings (and potentially just from one NIC) are merged: a requested poll task is placed in the poll list and processed in a round robin manner in the `net_rx_action()` function. By avoiding any overhead, when merging these ingress paths the NAPI ensures an important design objective: i.e. "robustness at any input rate and any number of input devices" [6]. Nevertheless, significant effects may e.g. be introduced by the increased number of available `sk_buffs` and `rx_rings` that lead to additional IRQ overhead.

E. Side Effects

As we have described the amount of ingress NICs that provide packets via the Linux NAPI should not affect the maximum packet rate of a software router significantly. In reality NIC vendors have implemented further techniques to increase the performance, e.g. Intel's interrupt throttling rate [24] that allows to delay interrupts by specifying a static or dynamic maximal rate of interrupts per second. As this maximal rate is ensured by delay and coalescing of interrupts in case of higher rates of ingoing packets these techniques are generally referred to as interrupt coalescing (IC). IC techniques work on a per-NIC-port-basis (or even per `rx_ring`) and therefore *may* introduce measurable effects of multiple NICs. As these are vendor specific features and not part of each NIC driver we do not analyze them further in this paper.

IV. Measuring the Impact of Multiple NICs

In this section we explain our measurements to quantify the overhead caused by distributing the load across multiple NICs in a software router. After defining our measurement goals, we derive a suitable measurement environment. We split our detailed description of the measurement environment in two parts, describing the load generator and the software router configuration.

A. Measurement Goals

The primary goal of our measurements is to assess the overhead of using multiple NICs for packet processing in a software router. Thus, we set up a Linux-based software router on a dedicated machine. We then test the routing performance of the software router in different load scenarios and measure the maximum packet rate of the device for each of the scenarios. The maximum achievable packet rate is suitable as an indicator for performance overhead, as it indicates the efficiency of packet processing. The higher the maximum observed packet rate is, the lower is the processing overhead of the software router.

As we are interested in both the resulting performance, and the root causes for overhead when using multiple NICs, we perform white-box measurements and look into our software router during the tests. First, we capture the CPU utilization on

the software router in each of the scenarios. This allows us to compare the CPU load when using a different number of NICs and to calculate the number of cycles per packet necessary in each scenario, which is a direct indicator of the processing efficiency. A lower observed CPU load at a fixed packet rate indicates a lower processing overhead in this scenario.

In Section III we gave an overview of the packet reception and transmission process in the Linux NAPI. We present that the packet processing mechanisms heavily rely on interrupt handling and the avoidance of interrupts in the first place. Therefore, the number of interrupts generated by the NICs is a key parameter when analyzing the performance overhead by multiple NICs. We measure the number of interrupts on the Linux router to test this hypothesis.

B. Measured Scenarios

In order to measure the overhead of using additional NICs for packet processing, we create measurement scenarios where we keep all parameters stable except the number of NICs. Our hypothesis is that at a fixed amount of traffic the processing overhead changes with the number of NICs involved in the processing. Basically, this means that we keep the total amount of traffic sent to the software router stable, but split it in equal parts among a varying number of tested NICs. We derive three scenarios using 2 – 4 NICs as illustrated in Figure 2.

The scenario utilizing two NICs (Fig. 2(a)) utilizes bidirectional forwarding between two different NICs, splitting the load equally among both interfaces. The scenarios involving three and four NICs (Fig. 2(b) and 2(c)) involve a circular forwarding also splitting the load equally among all interfaces involved.

We test the maximum throughput of the software router according to RFC 2544 [25], which defines requirements for device benchmarking tests. Our software router, the *device under test* (DUT), is connected to a machine that generates packets for the DUT and captures packets coming back from the device.

Packet processing on the DUT is limited to one specific core in all our tests. For multicore-scaling we refer to our previous work that shows that the maximum possible packet rate scales linearly with a limited number of cores [5].

C. Software Router Configuration

Hardware We set up the software router on an off-the-shelf rack server. The machine comes equipped with a dual-socket Supermicro X9DRH-iTF Mainboard, two Intel Xeon E5-2640V2 CPUs with 8 cores each running at a maximum clock speed of 2.00 GHz. The mainboard comes with a dual-port 10 Gigabit Ethernet NIC on board (Intel Ethernet Controller X540). We added another dual-port 10 Gigabit Ethernet NIC (Intel 10 Gbit Network Adapter X540-T2) using the same controller. In total this makes four 10 Gbit/s NICs that we use for our tests. All four NICs are attached to the first of the two CPUs on board which is important because transferring packets from one CPU to the other via the dual-IO-hub would result in decreased performance [2].

CPU Settings When setting up and configuring the software, our goal is to make the behavior of the DUT predictable to get repeatable measurements and end up with meaningful results. We therefore try to eliminate side-effects that influence the performance of our software router. First, we disable the *Turbo Boost* and dynamic frequency underclocking features, which set the CPU clock speed in a non-predictable way. This pins the CPU clock speed to the maximum rate of 2 GHz during all tests. We also disable *Hyper-Threading*, which schedules virtual cores onto physical cores in an unpredictable way.

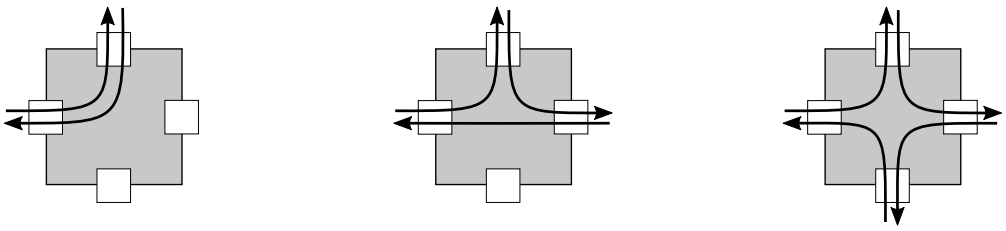
NIC Settings The NICs also need to be configured to facilitate reliable measurements. We disable Ethernet flow control, a feature that limits the network traffic in case one system is overloaded, because we also want to measure the DUT in overload situations. As multi-core measurements are out of scope, we configure only one `rx_ring` and `tx_ring` per NIC port. On top of that we disable the proprietary interrupt throttling feature offered by Intel, as explained in Section III. The Rx/Tx ring buffer sizes are set to 512 descriptors and offloading features such as *Large Receive Offload*, *Large Segment Offload*, and *Checksum Offloading* are disabled.

Software Our test systems run grml-Linux, a Debian-based live system distribution, running kernel version 3.7. As we want to measure the forwarding performance of the Linux-specific routing stack, we enable *ip_forward* and set static routes. As stated in the measurement goals, we configure the router to process all packets on one designated core. We already configured one `rx_ring` and `tx_ring` for each NIC port, now we bind the interrupts from all these queues to the same CPU core. We explicitly disable *irqbalance*, which has the goal to balance high-volume interrupts equally across all cores. Finally, we make sure the router can process incoming packets instantly without requiring any lookups, especially the ARP table is set manually before testing. In general we try to avoid any cross-traffic, e.g. by using statically assigned IP addresses.

System Performance Indicators We measure the CPU load and interrupt rate using `perf stat`, a lightweight tool to gather performance counters. To get reliable results, we delay launching perf stat for three seconds and let it end early before the end of a test run, to make sure that the system is fully loaded during the measured interval. We did not observe an influence of the measured throughput when running `perf stat`, in contrast to running a full-blown profiling setup, which would impact the system performance noticeably.

D. Measurement Device Configuration

We further limit the degrees of freedom in our measurement by using uniformly-sized packets with an Ethernet frame length of 64 Bytes, the minimum frame length supported by Ethernet. We previously showed that the performance of Linux-based software routers scales with the packet rate (number of packets per second) independent from the frame size as long as the link bottleneck is not reached [5].



(a) 2 NICs (b) 3 NICs (c) 4 NICs

Fig. 2. Overview of the traffic flow in the three measured scenarios. The gray box signifies the software router, while the arrows represent the traffic flows.

Load Generator To make use of our 10 GBit/s NICs we use the *zsend* software load generator, based on the packet processing framework *PF_RING ZC* [18]. We configure zsend to generate a constant bit rate (CBR) stream of uniform-sized UDP packets. As we want to distribute the load equally among several interfaces, we run zsend for each outgoing interface with a fraction of the specified packet rate, so that the total load generated sums up to the target packet rate. Additionally, we set the destination IP address, specifically for each instance of zsend, so that we can control the output interface on the DUT.

Packet Counters Our NICs feature hardware packet counters that accurately count the number of packets transmitted and received. We periodically poll these hardware registers to keep track of the number of packets sent and received on each network interface.

V. Performance Evaluation with Simulations

The methodology of simulations is often used in education and scientific studies because the setup of real testbed which implies complex configurations of the networking devices and links is often expensive and time consuming. Instead, researchers can use simulators as a cost-effective approach to design, validate, and analyze new ideas and optimizations in a controlled and reproducible manner.

A. Modeling of Resource Management in ns-3

To model a Linux software router, we apply our general modeling approach for intra-node resource management in resource-constrained network nodes [12]. Among others, this modeling approach introduces the terms resource manager, task unit, resource, and resource pool. A *resource manager* is an abstraction of the fundamental functions of the operating system (e.g. resource allocation, scheduling). All resources of a specific type are located in a resource pool which is managed by the resource manager according to a specific scheduling strategy (e.g. round-robin, priority scheduling). A *task unit* (TU) models an OS process or thread which requires specific resources like a CPU core or memory to be executed. Therefore, if a task unit has to process packets, it has to request the corresponding resources. For instance, in case of the Linux NAPI this refers to an IRQ after the reception of a packet. If the requested resource is available, then the resource manager sends a reply to the task unit which corresponds to the scheduling of an Rx ring by the NAPI.

B. Modeling of a Linux Software Router

Our Linux software router model consists of multiple network interface card ports ($NIC_0, \ldots, NIC_n$) and multiple CPU cores ($C_0, \ldots, C_k$) as depicted in Fig. 3. In the network simulator ns-3 each NIC port is represented by a specific *Net Device*. The *Net Device Classifier* assures a one-by-one mapping between net devices and its corresponding net device task units of the ns-3 resource-management module to model the ingoing NIC port ($TU\,NIC_{in}$). A net device task unit models the behavior of the network card controller. For instance, the multi-queue feature of modern NICs is modeled in that way that the incoming traffic may be classified based on specific packet attributes into one of the incoming queues (aka. Rx rings) of the successor task units to achieve traffic load balancing or prioritization of specific traffic.

In a Linux software router, these successor task units model behavior of the interrupt moderation of the Linux NAPI. There may be multiple Rx rings which are served by one NAPI thread pinned to a specific CPU core. This NAPI thread is modeled as multiple NAPI task units ($TU\,NAPI$). Each NAPI task unit possesses a dedicated incoming queue which models a specific Rx ring. Therefore, to model a Linux software router with $(n+1)$ NICs and $(k+1)$ CPU cores we need $(n+1)\cdot(k+1)$ NAPI task units. These NAPI task units compete for the same shared CPU core resource. Thus, NAPI task units which are pinned to the same core cannot be processed simultaneously. This refers to the fact that a NAPI thread in the real system can only serve one Rx ring at a certain time.

Depending on the type of the configured packet processing of the software router, diverse task units for each packet processing step may follow. For instance, in the case that the software router models a VPN gateway then a processing task unit ($TU\,Process$) is included which represents the packet processing step for IPsec encryption. In consequence, each additional processing task unit requires CPU resources ($C_0, \ldots, C_k$) which leads to an increase of the required CPU cycles per packet. After the actual packet processing, the packet is enqueued in the corresponding task unit which models the outgoing network interface card ($TU\,NIC_{out}$). There is a one-by-one mapping between the task units for the outgoing NIC ports and the corresponding ns-3 net devices. Finally, the packet is enqueued in the transmission queue of the outgoing net device and the processing of the native ns-3 core continues.

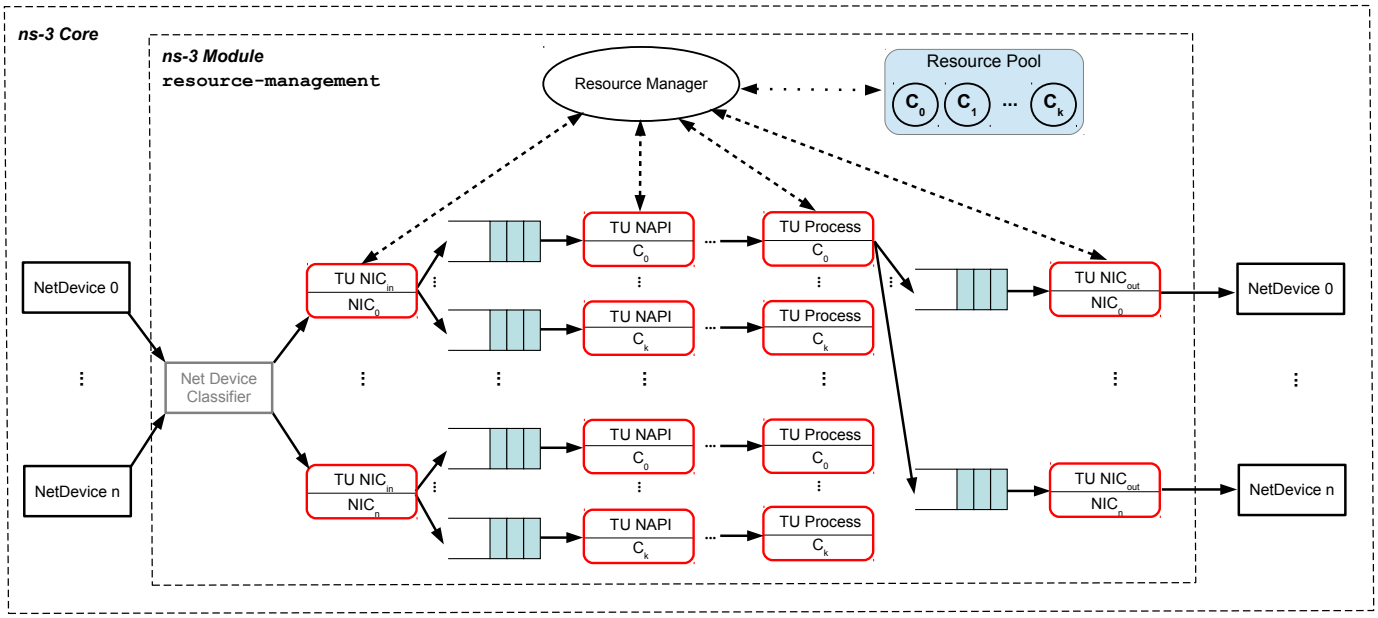


Fig. 3. Node model of a Linux software router with multiple network cards

C. Model Assumptions and Limitations

The software router model is subject to the following assumptions and limitations.

- The simulation model assumes that the CPU constitutes the bottleneck during the packet processing. Furthermore, we assume a linear scaling in the throughput when applying multiple CPU cores in parallel. This behavior was also shown by us [5] and other researchers [1].
- The simulation model considers modern NICs with multi-queue support which means that each CPU core has at least one Rx/Tx ring per NIC. This assures that every CPU core is able to exclusively access its dedicated Rx/Tx ring without resource contention. Thus, the modeling of locking mechanisms to access the Rx/Tx rings can be omitted. Besides, the processing of a specific packet always remains at the same CPU core to avoid cache misses which aligns with common recommendations for parallel packet processing in multi-core PC systems [1].
- We assume that the outgoing link is not the bottleneck. Therefore, the model does not implement any queuing disciplines (qdisc) at the egress NIC. This is in contrast to many network simulators like ns-3 [26] which assume the outgoing link to be bottleneck.
- The model neglects concurrent processes or threads which may lead to context switches. However, concurrent processes can be easily modeled by introducing additional task unit which apply for the same shared CPU resource. Furthermore, the OS scheduling overhead for managing multiple processes is neglected.
- Additional latencies introduced by other hardware components (e.g. DMA, PCI) are omitted but can be modeled with the help of our ns-3 resource-management extension to set up more fine-grained case studies.

VI. Case Study: Scaling of Linux Software Routers

In this section we evaluate the influence of the number of network cards with respect to the packet processing performance of a Linux software router. Firstly, we measure the performance of a Linux software router in real testbed experiments. Based on these testbed measurements we calibrate and validate our Linux software simulation model which is based on our *resource management* extension module for the network simulator ns-3.

A. Case Study Scenario

1) Network Topology: The case study scenario consists of a specific number of end systems connected to a Linux software router which acts as the device under test. The number of end systems is varied to investigate the influence of multiple network cards (cf. Fig. 2).

2) Load Generation: For all conducted measurements the load generation is based on the following conditions.

- All traffic is mapped to only one CPU core of the Linux software router because earlier research showed that the performance of multi-core CPUs linearly scales with the number of CPU cores [2], [3], [5], [27].
- The offered load is uniformly distributed between the number of NICs of the software router.
- All measurements are conducted with an Ethernet frame length of 64 B, because we have already shown that the packet size has no significant impact on the performance in case of IP routing. Furthermore, with small packet sizes we prevent that the network interface cards become the bottleneck at high offered loads.

3) Linux Software Router Configuration:

- The considered CPU core of the software router runs at a clock frequency of 2 GHz. The Intel features turbo boost and hyper-threading are disabled to prevent the system from tuning the clock frequency because this would evidently influence the performance of the software router.
- All used network cards are 10 Gbps Ethernet NICs.
- The interrupt moderation of the Linux software router relies on the NAPI. Therefore, we disabled the proprietary Intel NIC feature ITR (Interrupt Throttling Rate) [28], which is an additional interrupt moderation feature to avoid side effects.

As explained in Section IV we measured the packet forwarding performance of a real Linux software router. These testbed measurements aim to serve as calibration points for our software router model described in Section V. We therefore use the same parameters for all three scenarios during measurement and simulation. All measurements were conducted with CBR traffic consisting of uniform UDP packets with an Ethernet frame length of 64B. We present the results of these measurements in Figure 4(a) and Figure 4(b).

B. Model Calibration and Validation

Our Linux software router model was calibrated and validated for IP packet processing based on real testbed measurements of a Linux Software Router. The testbed measurements were conducted based on constant bit rate (CBR) traffic. The measured as well as the calibrated values for the simulation model are shown in Fig. 4(a) and 4(b). We measured a maximum throughput of the Linux software router of 0.9 Mpps for a single CPU core running at 2 GHz which relates to a service time of 1111 ns resp. 3667 CPU cycles per packet.

Through internal measurements by profiling the Linux networking stack with the tool *perf*, we measured the effort for the packet processing in detail. We observed that the service time is nearly constant and independent of the packet size because each IP packet requires a similar number of CPU cycles for the IP header processing (e.g. routing table lookup). The profiling results show that ca. 98.5 % of these 1111 ns are consumed for the polling of packets which consists of 86 % resp. 940 ns for the packet reception and actual packet processing (e.g. NAPI, IP processing) as well as 14 % resp. 154 ns for the packet transmission (e.g. Tx ring cleaning).

Furthermore, based on the testbed measurements we estimate the cost for an ISR with ca. 3400 ns which arises if an incoming packet causes an IRQ. Besides, we use a NAPI poll size of 64 and a Tx work limit of 256 which refer the defaults of the Linux NAPI. Further details of the derivation of the model calibration parameters are presented in [13].

The simulation results coincide with the testbed measurement values very well which indicates that our simulation model is valid and has been implemented correctly (Fig. 4). For the CPU utilization, a derivation offset of 10 to 20 % occurs for offered loads between 0.2 and 0.4 Mpps due to additional processing costs which are not considered in the simulation model. Furthermore, the applied CBR traffic cannot be precisely guaranteed with the load generator in the testbed. This is in contrast to the load generation in the simulator which is able to assure accurate packet interarrival times.

C. Simulation Results

The case study scenario is modeled based on the network simulator ns-3 as well as our ns-3 extension module *resource-management* [12]. The end devices and the software router are represented as dedicated ns-3 nodes. For the load generation, on each end system a UDP client as well as a UDP server application is installed. The UDP clients generate traffic with a constant packet size of 64 B according to a Poisson stream. The Linux software router possesses limited resources, thus our ns-3 resource management module must only be deployed to the software router.

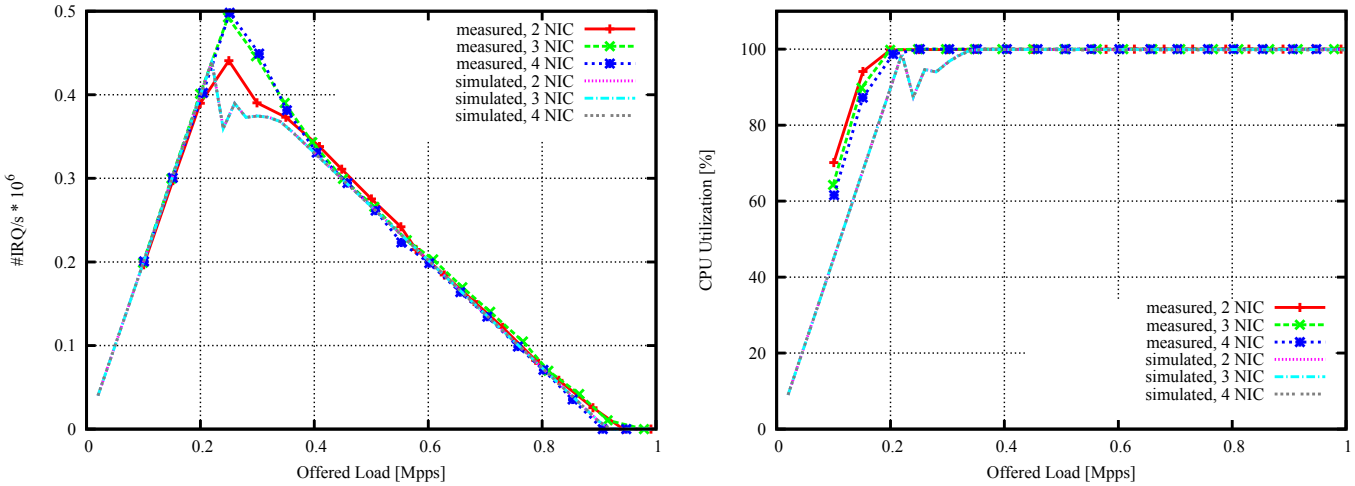
In contrast to the testbed measurement in the model calibration, the offered load is based on Poisson traffic with exponentially distributed interarrival times. Thus, the offered traffic is more realistic because short packet bursts as well as short idle times with no packet arrivals occur. Besides, for a specific offered load the overall IRQ rate is smaller with Poisson traffic in comparison to CBR traffic because when multiple packets arrive in a short burst only one IRQ is triggered which also reduces the CPU utilization. We analyzed the IRQ rate, the CPU utilization, and the mean packet latency for a Linux software router with different numbers of NICs.

1) IRQ Rate and CPU Utilization: The IRQ rate is a measure for the IRQ overhead in number of IRQs in a specific time interval triggered by the NICs. The CPU utilization is a dimensionless metric for the usage of the CPU as the relation between the busy time and the total time of observation.

Fig. 5(a) and Fig. 5(b) show the offered load in million packets per second (Mpps) on the x-axis. Fig. 5(a) illustrates the IRQ rate in number of million IRQs per seconds on the y-axis whereas Fig. 5(b) represents the corresponding CPU utilization in percentage on the y-axis. Each of the lines in both figures represents a different configuration of the software router with respect to the number of NICs.

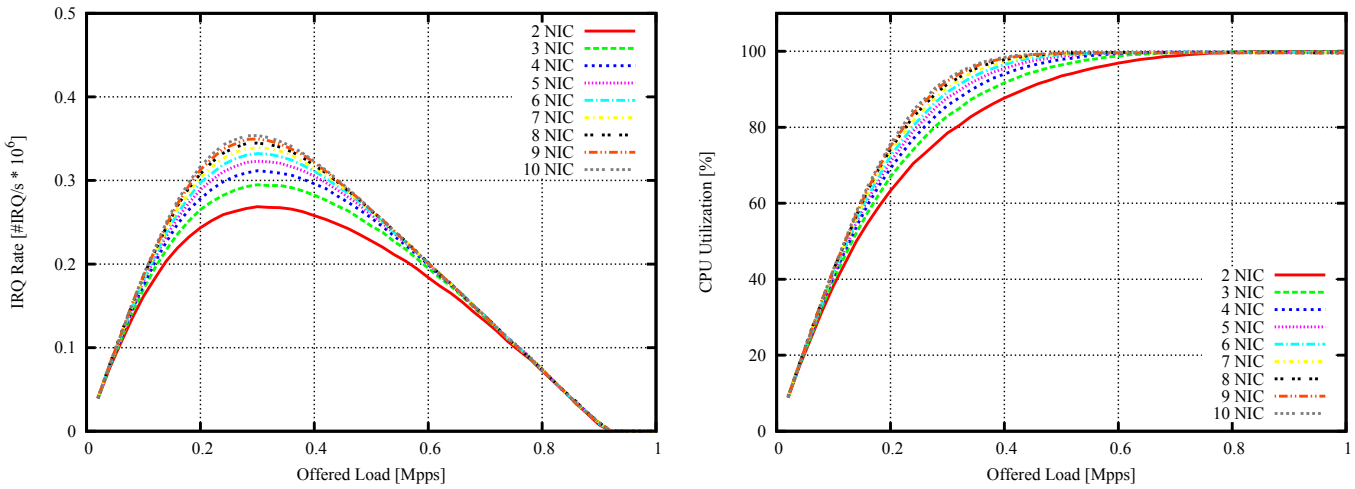
For low offered loads below 0.3 Mpps the CPU core is idle most of the time. Here, the Linux NAPI behaves interrupt-driven because almost every incoming packet causes an IRQ and is directly processed. Thus, when the offered load increases the IRQ rate as well as the CPU utilization linearly increases. However, the higher the offered load becomes the bigger is the difference in the IRQ rate and CPU utilization for the different number of NICs. For instance, at an offered load of 0.3 Mpps we observe an IRQ rate respectively CPU utilization with 2 NICs of $0.27 \cdot 10^6$ IRQ/s resp. 79 % whereas with 4 NICs the values are $0.31 \cdot 10^6$ IRQ/s resp. 86 %.

For an offered load higher than 0.3 Mpps, the IRQ rate decreases because there are often packets in the Rx rings of NICs and thus the Linux NAPI disables the IRQs for the corresponding Rx/Tx rings. Therefore, the Linux NAPI often works in poll-driven mode which saves IRQs. Thus, the rising of the CPU utilization also diminishes. When the offered load reaches the maximum throughput of the CPU core at ca.



(a) IRQ Rate (b) CPU Utilization

Fig. 4. Measured and simulated IRQ rate and CPU utilization for different numbers of NICs based on CBR traffic used for model calibration



(a) IRQ Rate (b) CPU Utilization

Fig. 5. Simulated IRQ rate and CPU utilization for different numbers of NICs based on Poisson traffic

0.9 Mpps, then all IRQs of the Rx rings are disabled to use the saved CPU cycles for the actual packet processing.

We conclude that with increasing offered loads the CPU core becomes saturated earlier with a higher number of NICs because at the same offered load each NIC triggers a separate IRQ for each packet. In contrast, with a smaller number of NICs at the same offered load, an IRQ serves multiple packets which leads to less IRQ-induced CPU utilization.

2) Packet Latency: The packet latency represents the delay which a packet incurs during its traversal through the software router. The packet latency consists of waiting and services times in several system components. In general, the waiting time of a specific packet depends on the number of packets prior to that packet in the waiting queue which is here an Rx ring. In the case of IP routing, the service time of a packet is constant (cf. Section VI-B). The packet latency is dominated by the waiting and service time at the CPU bottleneck.

Fig. 6 shows the offered load in Mpps on the x-axis and the simulated mean packet latency in microseconds on the y-axis. The mean packet latency is stated with 95 % confidence interval. Each of the lines represents a different configuration of the software router with respect to the number of NICs. The mean packet latency exponentially increases when the offered load increases. For offered loads higher than the maximum throughput of 0.9 Mpps of this router configuration, the mean packet latency is no longer well-defined. This is because arriving packets often hit a full incoming packet queue and must be dropped. For the same offered load, the mean packet latency is bigger the higher the number of NICs is. This is because with multiple NICs more IRQs must be handled which CPU cycles cannot be used for the actual packet processing which leads to higher waiting times. Consequently, with a higher number of NICs the full CPU utilization of 100 % is already reached for lower offered loads (cf. Figs. 4(b), 5(b)).

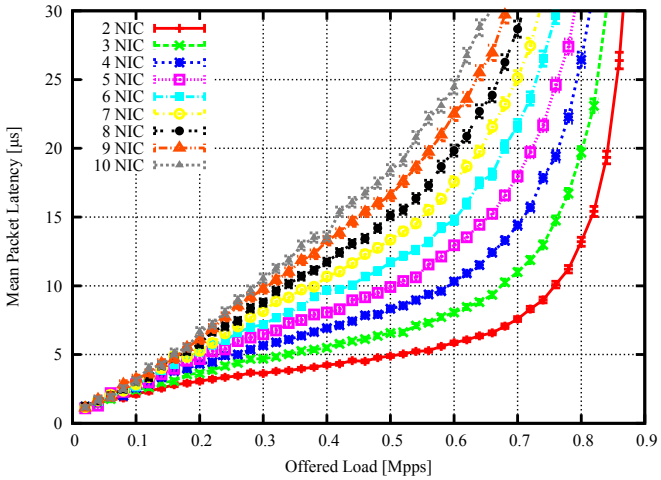


Fig. 6. Simulated mean packet latency for different numbers of NICs

VII. Summary and Outlook

We investigated the impact of multiple NICs for the performance of Linux software routers. We proposed a model of a Linux software router which we calibrated and validated based on testbed measurements. We showed that the Linux NAPI is able to scale with multiple NICs. However, the number of NICs have an impact on the performance, especially for the mean packet latency due to the additional IRQ overhead.

In future, we plan to further refine our model which includes the consideration of the latency caused by other system components. Besides, we are implementing a Linux NAPI extension for QoS-aware packet processing in software.

Acknowledgments

This research has been supported by the *Deutsche Forschungsgemeinschaft* (DFG; German Research Foundation) as part of the *MEMPHIS* project (GZ: WO 722/6-1). We would like to acknowledge the valuable contributions through numerous in-depth discussions from our colleagues Alexander Beifuß, Paul Emmerich, Sebastian Gallenmüller, Andrey Kolesnikov, and Dr. Klaus-Dieter Heidtmann.

References

[1] M. Dobrescu, N. Egi, K. Argyraki, B. Chun, K. Fall, G. Iannaccone, A. Knies, M. Manesh, and S. Ratnasamy, "RouteBricks: Exploiting Parallelism To Scale Software Routers," in *ACM Symposium on Operating Systems Principles (SOSP)*, October 2009.

[2] S. Han, K. Jang, K. Park, and S. Moon, "Building a Single-Box 100 Gbps Software Router," in *IEEE Workshop on Local and Metropolitan Area Networks (LANMAN)*, May 2010, pp. 1–4.

[3] K. Fall, G. Iannaccone, M. Manesh, S. Ratnasamy, K. Argyraki, M. Dobrescu, and N. Egi, "RouteBricks: Enabling General Purpose Network Infrastructure," *ACM SIGOPS Operating Systems Review*, vol. 45, no. 1, pp. 112–125, 2011.

[4] R. Bolla and R. Bruschi, "PC-based Software Routers: High Performance and Application Service Support," in *ACM SIGCOMM Workshop on Programmable Routers for Extensible Services of Tomorrow (PRESTO)*, August 2008, pp. 27–32.

[5] T. Meyer, F. Wohlfart, D. Raumer, B. E. Wolfinger, and G. Carle, "Validated Model-Based Performance Prediction of Multi-Core Software Routers," *Praxis der Informationsverarbeitung und Kommunikation (PIK)*, vol. 37, no. 2, pp. 93–107, 2014.

[6] J. H. Salim, R. Olsson, and A. Kuznetsov, "Beyond Softnet," in *5th Annual Linux Showcase & Conference*, vol. 5, 2001, pp. 18–18.

[7] J. H. Salim, "When NAPI Comes to Town," in *Linux Conference*, 2005.

[8] K. Salah and M. Hamawi, "Performance of IP-Forwarding of Linux Hosts with Multiple Network Interfaces," *Journal of Network and Computer Applications*, vol. 36, no. 1, pp. 452 – 465, 2013.

[9] R. Chertov, S. Fahmy, and N. Shroff, "A Device-Independent Router Model," in *IEEE Conference on Computer Communications (INFOCOM)*, April 2008.

[10] K. Salah, K. El-Badawi, and F. Haidari, "Performance Analysis and Comparison of Interrupt-Handling Schemes in Gigabit Networks," *Computer Communications*, vol. 30, no. 17, pp. 3425–3441, 2007.

[11] K. Salah, "Modeling and Analysis of PC-based Software Routers," *Computer Communications*, vol. 33, no. 12, pp. 1462–1470, 2010.

[12] T. M. Runge, B. E. Wolfinger, S. Heckmüller, and A. Abdollahpouri, "A Modeling Approach for Resource Management in Resource-Constrained Nodes," *Journal of Networks*, vol. 10, no. 01, pp. 39–50, 2015.

[13] A. Beifuß, D. Raumer, P. Emmerich, T. M. Runge, F. Wohlfart, B. E. Wolfinger, and G. Carle, "A Study of Networking Software Induced Latency," in *International Conference on Networked Systems (NetSys)*, March 2015.

[14] P. Emmerich, D. Raumer, F. Wohlfart, and G. Carle, "A Study of Network Stack Latency for Game Servers," in *13th Annual Workshop on Network and Systems Support for Games (NetGames'14)*, Nagoya, Japan, Dec. 2014.

[15] ——, "Performance Characteristics of Virtual Switching," in *IEEE International Conference on Cloud Networking (CloudNet)*, Luxembourg, October 2014.

[16] R. Bolla and R. Bruschi, "Linux Software Router: Data Plane Optimization and Performance Evaluation," *Journal of Networks*, vol. 2, no. 3, pp. 6–17, June 2007.

[17] M. Dobrescu, K. Argyraki, and S. Ratnasamy, "Toward Predictable Performance in Software Packet-Processing Platforms," in *USENIX Conference on Networked Systems Design and Implementation (NSDI)*, April 2012.

[18] F. Fusco and L. Deri, "High Speed Network Traffic Analysis with Commodity Multi-core Systems," in *Internet Measurement Conference*, November 2010, pp. 218–224.

[19] A. Botta, A. Dainotti, and A. Pescapé, "Do You Trust Your Software-based Traffic Generator?" *IEEE Communications Magazine*, vol. 48, no. 9, pp. 158–165, 2010.

[20] P. Emmerich, F. Wohlfart, D. Raumer, and G. Carle, "MoonGen: A Scriptable High-Speed Packet Generator," *ArXiv e-prints*, Oct. 2014.

[21] S. Gallenmüller, P. Emmerich, F. Wohlfart, D. Raumer, and G. Carle, "Comparison of Frameworks for High-Performance Packet IO," in *ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS 2015)*, 2015.

[22] T. M. Runge, A. Beifuß, and B. E. Wolfinger, "Low Latency Network Traffic Processing with Commodity Hardware," in *International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS)*, July 2015.

[23] D. Raumer, F. Wohlfart, D. Scholz, P. Emmerich, and G. Carle, "Performance Exploration of Software-based Packet Processing Systems," in *Leistungs-, Zuverlässigkeits- und Verlässlichkeitsbewertung von Kommunikationsnetzen und verteilten Systemen, 8. GI/ITG-Workshop MMBnet*, September 2015.

[24] P. Emmerich, D. Raumer, A. Beifuß, L. Erlacher, F. Wohlfart, T. M. Runge, S. Gallenmüller, and G. Carle, "Optimizing Latency and CPU Load in Packet Processing Systems," in *International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS)*, 2015.

[25] S. Bradner and J. McQuaid, "Benchmarking Methodology for Network Interconnect Devices," RFC 2544 (Informational), Internet Engineering Task Force, March 1999.

[26] T. R. Henderson, M. Lacage, G. F. Riley, C. Dowell, and J. Kopena, "Network Simulations with the ns-3 Simulator," *ACM SIGCOMM Demonstration*, August 2008.

[27] S. Han, K. Jang, K. Park, and S. Moon, "PacketShader: A GPU-Accelerated Software Router," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4, August 2011.

[28] Intel Corporation, "Interrupt Moderation Using Intel GbE Controllers," April 2007.

Koexistenz nicht-kompatibler TCP-Varianten in softwaredefinierten Netzen

Mario Hock
Institut für Telematik
Karlsruher Institut für Technologie
E-Mail: mario.hock@kit.edu

Mario Willems
Institut für Telematik
Karlsruher Institut für Technologie
E-Mail: mail@mariowillems.de

***Zusammenfassung*—Die im Internet eingesetzten Staukontrollverfahren sind überwiegend verlustbasiert. Das heißt, dass der reguläre Betrieb dieser Mechanismen vorsieht, Router-Warteschlangen bis zum Überlauf zu füllen (bzw. bis der Router gezielt beginnt Pakete zu verwerfen). Dies führt zu Paketverlusten und hohen Latenzen. Anwendungen wie hoch qualitative Videokonferenzen und moderne Cloud-Dienste sind allerdings auf geringe Latenzen, und somit auf kurze Warteschlangen angewiesen. Zu deren Unterstützung wären Staukontrollverfahren vorteilhaft, die ohne lange Warteschlangen auskommen. SDN-Netze können dabei helfen dieses Problem zu überwinden. Diese Arbeit untersucht inwieweit OpenFlow-/SDN-Switches dazu genutzt werden können, den Verkehr verschiedener TCP-Varianten voneinander zu trennen und somit die Anforderung nach Kompatibilität zwischen diesen Varianten aufzuheben. Experimentell wurde nachgewiesen, dass inkompatible TCP-Varianten dadurch separiert werden können, dass sie auf physisch getrennte Pfade umgeleitet werden. Stehen keine getrennten Pfade zur Verfügung, können Rate-limiter, sog. „OpenFlow-Meter“, zur Separierung genutzt werden. Hierbei sind allerdings einige Einschränkungen zu beachten.**

I. Einleitung

Das Internet wird zunehmend für interaktive, datenintensive Anwendungen genutzt, wie z. B. hochqualitative Videokonferenzen oder das „Screensharing“ von HD-Monitoren. Solche Anwendungen benötigen geringe Latenzen bei der Datenübertragung, produzieren aber so große Datenmengen, dass sie nicht einfach gegenüber dem restlichen Internetverkehr priorisiert werden können, wie es heute z. B. bei der Internettelefonie (VoIP) der Fall ist. Während die Übertragungskapazitäten im Internet kontinuierlich steigen, fallen die Latenzen allerdings nicht in vergleichbarem Maße. Einerseits sind die Latenzen durch die physische Signalausbreitung nach unten beschränkt. Andererseits zeigt die *„Bufferbloat“*-Problematik [1], dass auch die Warteschlangenverzögerung in Routern und Switches einen erheblichen Anteil daran hat.

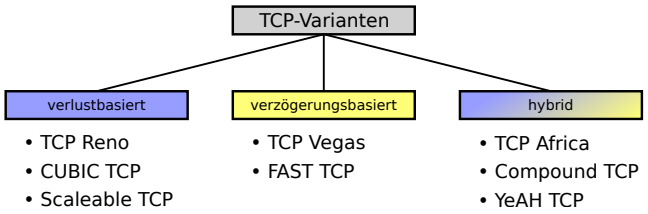
Durch die erhöhte Warteschlangenverzögerung erhöht sich auch die *Umlaufzeit (RTT)* und damit die vom Benutzer erfahrene Latenz. Durch volle Warteschlangen kommt es darüber hinaus häufig zu Paketverlusten. Laut [2] sorgt bei der Übertragung von Web-Inhalten ein Paketverlust dafür, dass sich die Übertragungszeit im Schnitt um das *Fünffache* erhöht. Die hohe Zahl kommt dadurch zustande, dass der Paketverlust häufig am Ende der Übertragung auftritt und daher erst verzögert erkannt wird.

Wünschenswert wäre daher eine Staukontrolle, die nur sehr wenige Paketverluste erzeugt und die Auslastung der Warteschlangen gering hält. Die in heute genutzten TCP-Varianten implementierten Staukontrollmechanismen erfüllen diese Anforderungen nicht. Alternative Verfahren, welche die Warteschlangenlänge gering halten, werden im Internet bisher nicht genutzt, da sie von den bestehenden TCP-Varianten unterdrückt werden. Diese Arbeit beschäftigt sich daher mit der Frage, wie das Netz Fairness zwischen inkompatiblen TCP-Varianten herstellen kann, sodass die gleichzeitige Nutzung alternativer Staukontrollverfahren ermöglicht wird.

Im Folgenden wird untersucht, welche Möglichkeiten in OpenFlow-basierten SDN-Netzen bestehen, um die Koexistenz inkompatibler TCP-Varianten zu ermöglichen.

II. Inkompatible TCP-Varianten

Da das Internet eine Stausituation nicht explizit an den Sender signalisiert[1], müssen die dort eingesetzten Staukontrollalgorithmen mit *impliziten* Stausignalen arbeiten. Hierbei werden meist *Paketverluste* als Stausignal genutzt. Sofern im Netz kein *Active Queue Managment* (wie z.B. RED [4]) genutzt wird, treten diese erst dann auf, wenn die Warteschlangen bereits überfüllt sind und das Netz Pakete verwerfen muss. Alternativ kann ein Stau auch über einen Anstieg der Verzögerung in Senderichtung, bzw. der Umlaufzeit, erkannt werden. Man unterscheidet daher zwischen TCP-Varianten mit *verlustbasierter*, *verzögerungsbasierter* und *hybrider* Staukontrolle.



Fairness zwischen konkurrierenden Verbindungen kann nur erreicht werden, wenn die genutzten Staukontrollalgo-

[1] Mittels dem in RFC 3168 [3] standardisierten *„Explicit Congestion Notification (ECN)“* kann das Netz den Sender zwar über einen Stau informieren, ECN wird allerdings im Internet nicht flächendeckend eingesetzt.

rithmen identisch oder zumindest zueinander *kompatibel* sind. RFC 2914 [5] definiert eine Verbindung als *„TCP-kompatibel“*, wenn sie bei Stau nicht mehr Daten ins Netz sendet als eine *TCP Reno*-Verbindung (RFC 2581 [6]). Eine *aggressivere* Verbindung würde die TCP-kompatiblen Verbindungen unterdrücken. Staukontrollalgorithmen, die ein solches Verhalten produzieren, sollen daher im Internet nicht genutzt werden. Ein Staukontrollalgorithmus wird als *aggressiv* bezeichnet, wenn er seine Senderate sehr schnell erhöht, oder sie während eines Stau nur leicht oder erst spät reduziert.

Verzögerungsbasierte TCP-Varianten (wie z. B. TCP Vegas [7]) erkennen einen drohenden Stau bereits sehr früh und reagieren *proaktiv* mit einer Reduktion der Senderate. Dies führt zu deutlich kürzeren Warteschlangen und zu einer sehr geringen Zahl an Paketverlusten. Konkurrieren sie jedoch mit den aggressiveren verlustbasierten Verfahren, bekommen sie keinen fairen Anteil an der Übertragungskapazität. Hybride Varianten (z. B. Compound TCP [8]) sind hingegen deutlich aggressiver als TCP Reno, solange sie keinen Anstieg der Verzögerung erkennen können. Steigt die Verzögerung über einen festgelegten Schwellenwert, reduzieren sie ihre Aggressivität auf das Niveau von TCP Reno, wodurch die Forderung nach TCP-Kompatibilität erfüllt wird. Gleichzeitig sorgen sie so dafür, dass sie einen fairen Anteil an den Übertragungskapazitäten bekommen. Erst bei einem Paketverlust reduzieren sie die Größe ihres Staukontrollfensters. Wie die verlustbasierten, füllen demnach auch die hybriden TCP-Varianten die Warteschlangen bis zum Überlauf.

Volle Warteschlangen führen zu einer merklichen Erhöhung der Umlaufzeit und häufigeren Paketverlusten. Dies beeinträchtigt *alle* Verbindungen, die durch diese Warteschlangen laufen. Eine weniger aggressive TCP-Variante ist ebenfalls von diesen Auswirkungen betroffen, auch wenn sie selbst nur gering zu der hohen Auslastung der Warteschlange beiträgt. Erhöht eine zurückhaltende Variante in einer solchen Konkurrenzsituation nicht die eigene Aggressivität, wird die zurückhaltende Variante von aggressiveren Varianten unterdrückt, wie in Abb. 1 veranschaulicht. In der Abbildung ist der Durchsatz und die RTT einer *TCP Reno*-Verbindung und einer *TCP Vegas*-Verbindung zu sehen, die miteinander konkurrieren. Da TCP-Kompatibilität im Internet über das Verhalten von TCP Reno definiert ist, dient es hier als Repräsentant einer für eine im Internet typische TCP-Verbindung. TCP Vegas hingegen ist ein rein verzögerungsbasiertes Verfahren, das versucht nur wenige eigene Pakete in den Warteschlangen zu halten und selbst kaum Paketverluste induziert. Das Experiment wurde in einem lokalen Testbett, bestehend aus zwei Sendern, einem Switch und einem Empfänger (vgl. Abb. 1a), durchgeführt. Die genaue Testumgebung wird in Abschnitt V-A nochmals detailliert beschrieben.

Von Beginn an fällt ein Großteil der zur Verfügung stehenden Übertragungskapazität auf die TCP Reno-Verbindung, die TCP Vegas-Verbindung hingegen belegt nur einen kleinen Teil der Gesamtkapazität (Abb. 1b). Die TCP Reno-Verbindung beginnt nun den Puffer an der gemeinsamen Engstelle zu füllen, was für beide Verbindungen eine linear steigende Umlaufzeit zur Folge hat (Abb. 1c). Nach etwa 50 s kommt es zu einem Paketverlust und TCP Reno reduziert das Staukontrollfenster, wodurch sich die Warteschlange leert und die Umlaufzeit wieder auf ca. 50 ms fällt. TCP Vegas erkennt dies und erhöht kurzzeitig die eigene Senderate, zieht sich aber sofort zurück, sobald die Warteschlange durch die TCP Reno-Verbindung erneut gefüllt wird. Es ist erkennbar, dass die Senderate der TCP Reno-Verbindung im Anschluss an einen Paketverlust schnell wieder steigt. Dies ist darin zu begründen, dass sich die Senderate aus dem Quotienten von Staukontrollfenster und RTT ergibt. Die Verringerung der RTT gleicht in diesem Experiment somit teilweise die Reduktion des Staukontrollfensters aus, was zu dem hier beobachtbaren schnellen Anstieg der Übertragungsrate führt.

Dieses Experiment zeigt, dass eine TCP-Variante, die eine geringe Auslastung der Warteschlange anstrebt, keinen fairen Anteil an der Übertragungskapazität bekommt, wenn sie an einer Engstelle mit den heute üblichen TCP-Varianten konkurriert. Durch die stark gefüllte Warteschlange an der Engstelle erhöht sich auch die Latenz der TCP Vegas-Verbindung, obwohl der Vegas-Algorithmus sicherstellt, dass nur wenige eigene Pakete in der Warteschlange enthalten sind.

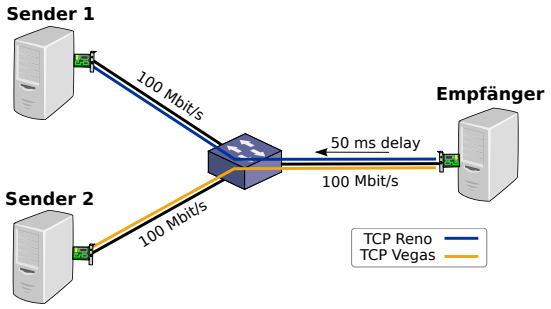
Im folgenden Abschnitt wird untersucht, welche Möglichkeiten in OpenFlow-basierten SDN-Netzen zur Verfügung stehen, um dieses Problem zu umgehen und die parallele Nutzung inkompatibler TCP-Varianten zu ermöglichen.

III. Lösungsmöglichkeiten mit OpenFlow

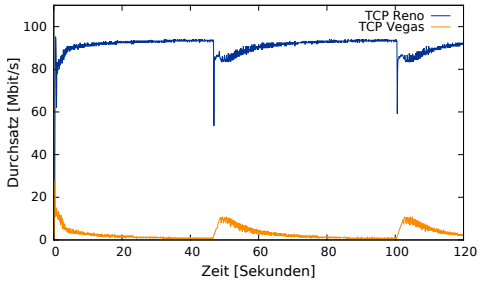
SDN-Netze unterscheiden sich insbesondere dadurch von klassischen Netzen, dass die Kontrollebene vom Datenpfad getrennt ist. Ein logisch zentraler *Controller* konfiguriert alle SDN-Switches in einem (Teil-)Netz. Dies ermöglicht ein besonders flexibles *Traffic Engineering*, das zur Separation inkompatibler TCP-Varianten genutzt werden kann. Stehen im Netz redundante Verbindungen zur Verfügung, kann der Controller inkompatible Verbindungen auf separaten Pfaden durch das Netz leiten. Stehen keine separaten Pfade zur Verfügung, können *Quality of Service (QoS)* Mechanismen genutzt werden, um auch auf einem gemeinsamen Übertragungsabschnitt die Fairness zwischen den Verbindungen zu verbessern. Diese Mechanismen sind im OpenFlow-Standard als *optional* gekennzeichnet und stehen daher nicht auf allen OpenFlow-Switches zur Verfügung.

Sowohl die Weiterleitungsregeln, als auch die QoS-Mechanismen werden vom Controller auf Basis von sogenannten *Flows* konfiguriert. Inkompatible TCP-Verbindungen müssen somit unterschiedlichen Flows zugewiesen werden. In einem Flow können die Datenpakete einer oder auch mehrerer TCP-Verbindungen zusammengefasst werden. Die Klassifikation findet dabei u. a. anhand von Feldern im IP- und TCP-Kopf statt.

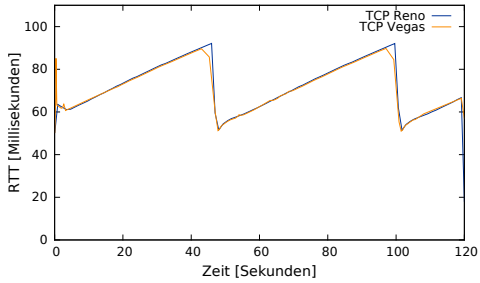
Um inkompatible TCP-Verbindungen unterschiedlichen Flows zuweisen zu können, muss dem Controller bekannt sein, welche TCP-Varianten von den entsprechenden Verbindung genutzt werden. Da bei TCP dem Kommunikationspartner nicht signalisiert wird, welche Variante genutzt wird,



(a) Testbett



(b) Durchsatz



(c) RTT

Abbildung 1: Ein TCP Vegas Sender wird von einem konkurrierenden TCP Reno Sender unterdrückt

kann der Controller diese Information nicht einfach beim Verbindungsaufbau ermitteln. Eine Klassifikation anhand des Sendeverhaltens, wie in [9], ist zwar generell möglich, aber sehr aufwändig und auch nicht sofort beim Verbindungsaufbau möglich, da die Verbindung eine Zeit lang beobachtet werden muss. Alternativ kann eine Anwendung, die eine inkompatible TCP-Verbindung aufbauen möchte und eine Sonderbehandlung durch das Netz wünscht, dies dem Controller signalisieren.

Da es in dieser Arbeit primär um die generelle Machbarkeit und die Untersuchung der Mechanismen geht, die eine parallele Nutzung inkompatibler TCP-Varianten ermöglichen, wurde hier der folgende Weg gewählt: Alle Verbindungen eines Endsystems nutzen (pro Netzwerkinterface) die gleiche TCP-Variante. Endsysteme, die eine inkompatible TCP-Variante nutzen, müssen sich zuvor beim Controller anmelden. Hierfür werden im Folgenden zwei *Verkehrsklassen* unterschieden: Loss und Delay

A. Verkehrsklassen

1) Die Klasse „Loss“ enthält die TCP-Varianten, welche die Warteschlange bis zum Überlauf füllen.
2) Die Klasse „Delay“ enthält die TCP-Varianten, die eine kurze Warteschlange anstreben.

In der Klasse Loss sind dementsprechend die heute im Internet üblicherweise genutzten verlustbasierten und hybriden TCP-Varianten enthalten. Da im Internet genutzte TCP-Varianten durch die Forderung der TCP-Kompatibilität zumindest bei Stau nicht aggressiver sein sollten als TCP Reno, werden in den Experimenten (Abschnitt V) grundsätzlich TCP Reno-Verbindungen als Vertreter dieser Klasse genutzt.

Die Klasse Delay ist insbesondere auf zukünftige TCP-Varianten ausgelegt, die einen hohen Durchsatz bei niedrigen Latenzen ermöglichen und nur wenige Paketverluste erzeugen. Bei Experimenten wird TCP Vegas als Vertreter dieser Klasse genutzt.

B. Mechanismen

Neben der Möglichkeit des *Traffic Engineering* bietet OpenFlow (ab der Version 1.3) zwei unterschiedliche *Quality of Service (QoS)* Mechanismen. Somit ergeben sich drei Mechanismen, die potentiell in der Lage sind die Koexistenz inkompatibler TCP-Varianten zu ermöglichen:

- Traffic Engineering
  1) Nutzung separater Pfade
- Quality of Service (QoS)
  2) Nutzung separater Warteschlangen mittels der *setQueue-Action*
  3) Nutzung von OpenFlow-*Meters*

1) Separate Pfade: Stehen in einem SDN-Netzwerk redundante Pfade zur Verfügung, können die Flows durch den Controller so programmiert werden, dass TCP-Verbindungen der Klasse Loss keine gemeinsamen Übertragungsabschnitte haben wie TCP-Verbindungen der Klasse Delay. Somit teilen sie sich keine gemeinsame Warteschlange und konkurrieren auch nicht um die verfügbare Übertragungskapazität am Ausgangsport. Daher ist keine Beeinträchtigung zwischen den Verbindungen zu erwarten. Dies wurde experimentell nachgewiesen und ist in Abschnitt V beschrieben.

Dieser Mechanismus kann mit allen OpenFlow-Switches genutzt werden, sofern redundante Pfade zur Verfügung

stehen. Ein Nachteil dieser Herangehensweise ist, dass die Ressourcen eines Pfads nur von der entsprechenden TCP-Variante genutzt werden können.

2) Separate Warteschlangen: Unterstützt der OpenFlow-Switch mehrere Warteschlangen pro Ausgangsport, können die Pakete eines Flows mittels der *setQueue-Action* in eine solche Warteschlange eingefügt werden. Die Warteschlangen müssen vorab auf anderem Weg (z. B. dem *OF-Config*-Protokoll [10]) konfiguriert werden.

Die Nutzung separater Warteschlangen ist konzeptionell ähnlich zur Nutzung separater Pfade. Inkompatible Verbindungen teilen sich auch hier keine gemeinsame Warteschlange und die verfügbare Übertragungskapazität kann fair auf die verschiedenen Warteschlangen aufgeteilt werden. Ein Vorteil gegenüber der Nutzung separater Pfade ist, dass Übertragungskapazität, die von einer der beiden Verkehrsklassen nicht genutzt wird, der anderen Verkehrsklasse zur Verfügung steht. Dies geschieht automatisch im Switch, ohne dass dafür eine Controller-Interaktion notwendig ist. Die *setQueue-Action* steht aber nicht auf allen OpenFlow-Switches zur Verfügung und konnte aus genau diesem Grund auch in dieser Arbeit noch nicht praktisch evaluiert werden. Eine experimentelle Untersuchung dieses Ansatzes ist aber für eine Folgearbeit geplant.

3) OpenFlow-Meters: Mit OpenFlow-*Meters* kann eine maximale Übertragungsrate für einen Flow oder eine Gruppe von Flows gesetzt werden. Dies kann genutzt werden um die Unterdrückung von weniger aggressiven TCP-Varianten durch verlustbasierte und hybride TCP-Varianten zu verhindern.

Anders als bei den beiden anderen Mechanismen werden die inkompatiblen Verbindungen hier nicht auf unterschiedliche Warteschlangen verteilt. Durch die Limitierung der Übertragungsrate werden die aggressiven Verbindungen aber effektiv daran gehindert die Warteschlange zu füllen. Freie Übertragungskapazität kann bei diesem Mechanismus nicht automatisch von den Flows einer anderen TCP-Variante genutzt werden, die Grenzen können aber vom Controller feingranular gesetzt und regelmäßig angepasst werden.

C. Weiterleitungsregeln

Es gibt zwei unterschiedliche Strategien, wie die Weiterleitungsregeln in SDN-Netzen festgelegt werden. Hierbei wird zwischen *Micro-Flows* und *Macro-Flows* unterschieden. Micro-Flows sind sehr spezifische Regeln, beispielsweise ein Flow pro TCP-Verbindung, oder pro kommunizierendem Endgerätepaar. Als Macro-Flows bezeichnet man allgemeinere Regeln, die viele potentielle Micro-Flows zusammenfassen, beispielsweise ein Flow zum Routing in ein anderes IP-Subnetz.

Alle drei Mechanismen können prinzipiell sowohl für Micro- als auch für Macro-Flows genutzt werden. Es muss jedoch sichergestellt sein, dass zwei inkompatible Verbindungen nicht innerhalb desselben Macro-Flows gruppiert werden.

IV. Umsetzung im SDN-Netz

Um neue Funktionen in ein SDN-Netz zu integrieren, muss der Controller erweitert werden. Dies wird üblicherweise in Form von *Apps* realisiert, die entweder über eine *Plug-in-Schnittstelle* Teil des Controller-Prozesses sein können oder externe Anwendungen sind, die mit dem Controller über eine sogenannte *Northbound-Schnittstelle* interagieren. Hierüber steht aber oft nur ein reduzierter Funktionsumfang zur Verfügung.

In einem realen SDN-Netz muss der Mechanismus *„Separate Pfade“* als Teil der Routing-App des Netzes implementiert werden. Zur Untersuchung der Funktionsfähigkeit des Mechanismus ist jedoch ein möglichst einfacher Testaufbau zu bevorzugen. Im Rahmen dieser Arbeit wurden daher statisch konfigurierte Flows genutzt, welche die beiden Verkehrsklassen über separate Pfade leiten.

Der Mechanismus *„OpenFlow-Meters“* muss dynamisch auf das Starten und das Beenden neuer Flows reagieren und die Meter-Begrenzungen dementsprechend anpassen. Daher wurde zur Untersuchung dieses Mechanismus eine App als Plug-in für den *OpenDaylight*-Controller entwickelt, die für die Programmierung der Flows und das Setzen von Meter-Begrenzungen zuständig ist.

A. Funktionsumfang der App

Das Einsatzgebiet der App beschränkt sich momentan auf Netze, die aus einem einzelnen OpenFlow-Switch und mehreren angeschlossenen Endsystemen bestehen. Die Flows werden durch die App nach einer *„SDN-Learning-Switch“*-Strategie programmiert. Das heißt, dass die App, ähnlich wie die Kontrollebene eines klassischen Switches, lernt welche Endsysteme an welchem Switchport angeschlossen sind. Möchten zwei Endsysteme miteinander kommunizieren, wird ein Flow mit den zugehörigen Sender- und Empfänger-MAC-Adressen programmiert und ggf. die Einstellungen der Meters angepasst (vgl. IV-B).

Es wird somit davon ausgegangen, dass ein Endsystem (pro Netzwerkschnittstelle) immer dieselbe TCP-Variante nutzt. Dies ist aber keine generelle Einschränkung der Mechanismen. Soll ein Endsystem mehrere inkompatible TCP-Varianten parallel nutzen können, müssen spezifischere Flows programmiert werden. Hierfür sind verschiedene Strategien denkbar. Beispielsweise könnten die Endsysteme unterschiedliche IPv6-Adressen für die beiden Verkehrsklassen nutzen. Anstelle von MAC-Adressen müssen die Flows in diesem Fall auf Basis von IP-Adressen programmiert werden, was über eine einfache Änderung an der App realisiert werden könnte.

Wie in Abschnitt III diskutiert, wird in dieser Arbeit davon ausgegangen, dass sich ein Endsystem, welches die Klasse Delay nutzen möchte, vorher beim Controller anmelden muss. Hierfür stellt die App eine *Rest-Schnittstelle* zur Verfügung.

B. Programmierung der Flows und Meter-Begrenzungen

Bevor der erste Flow programmiert wird, wird auf dem OpenFlow-Switch für jeden Switch-Port ein *Meter* konfiguriert. Dieser wird auf die maximale Übertragungsrate des

Ports gesetzt, sodass die Begrenzung effektiv deaktiviert ist. Dieses Vorgehen ermöglicht ein effizientes Setzen und Aufheben von dynamischen Begrenzungen für inkompatible Sender.

Empfängt der OpenFlow-Switch ein Datenpaket, das zu keinem Flow gehört, wird das Paket an den Controller weitergeleitet. Dies löst ein *„Packet-In"-Event* in der App aus, sodass diese einen passenden Flow programmieren kann.

Die Programmierung des Flows geht mit einer Anpassung der *Meter*-Einstellung einher. Wie in Abb. 2 dargestellt, wird die *Meter*-Begrenzung nur gesetzt, wenn sich Verbindungen beider Verkehrsklassen (Loss und Delay) einen Ausgangsport teilen. Somit wird keine Kapazität verschwendet, wenn gar keine Delay-Verbindungen existieren. Loss-Flows werden aber grundsätzlich an den (ggf. inaktiven) *Meter* gebunden. Kommt nun ein Delay-Flow hinzu, kann der *Meter* schnell umkonfiguriert werden, ohne dass alle Loss-Flows angepasst werden müssen.

Existieren sowohl Loss- als auch Delay-Flows, wird der *Meter* auf 50 % der Ausgangskapazität gesetzt. Eine Begrenzung in Abhängigkeit der *Anzahl* an Flows in den jeweiligen Klassen ist ebenfalls denkbar, aber in der App momentan nicht umgesetzt.

Die Flows sind mit einem *Timeout*-Wert versehen, sodass der OpenFlow-Switch einen Flow löscht, wenn dieser eine bestimmte Zeit lang nicht genutzt wurde. Der Switch benachrichtigt daraufhin den Controller. Analog zum Setzen neuer Flows überprüft die App dann, ob der *Meter* weiterhin benötigt wird, anderenfalls wird er wieder deaktiviert, indem er auf 100 % der Ausgangskapazität gesetzt wird.

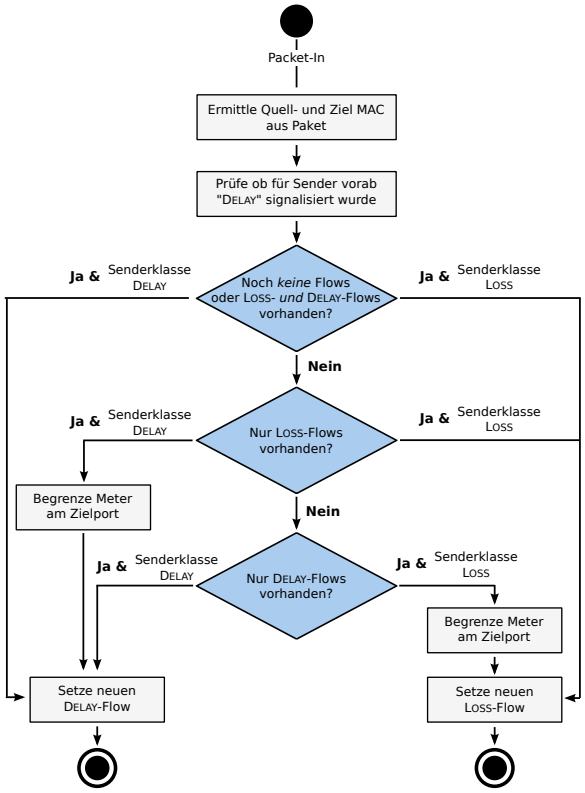


Abbildung 2: Setzen eines neuen Flows und Anpassung der *Meter*

V. Evaluation

In einem physischen Testbett wurden die beiden Ansätze

1) Nutzung separater physischer Pfade
2) Setzen dynamischer *Meter*-Begrenzungen mittels Controller-App

experimentell untersucht.

Alle Versuche wurden mehrfach wiederholt und zeigen reproduzierbare Ergebnisse. Aus Gründen der Übersichtlichkeit werden im Folgenden jeweils die Daten eines einzelnen Testlaufs dargestellt.

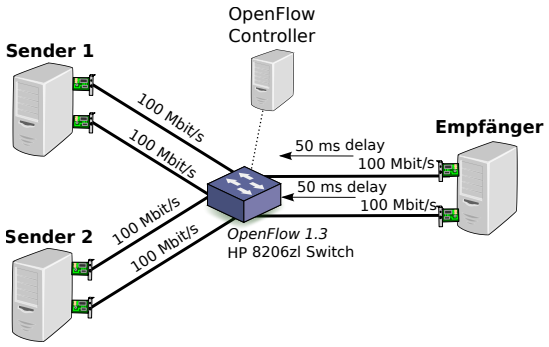
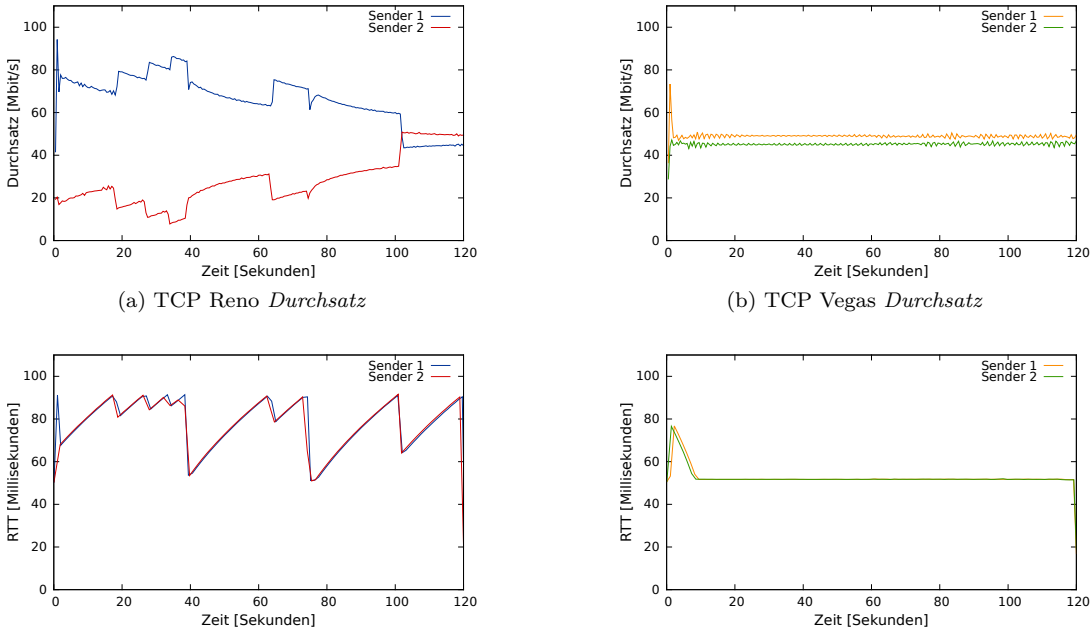


Abbildung 3: Testbett

A. Testbett

Das Testbett (Abb. 3) besteht aus einem OpenFlow-fähigen *HP 8206zl* Switch (Firmware-Version: K.15.15.0006, OpenFlow-Version: 1.3, nur v2 Module) und vier identischen Server-Rechnern, wovon einer als OpenFlow-Controller dient, einer als Empfänger und zwei als Sender. Die Sender und der Empfänger sind jeweils über zwei Netzwerkschnittstellen mit dem Switch verbunden. Die Server sind mit einer *Intel Xeon X3430 @ 2.40 GHz*-CPU und 12 GB-RAM ausgestattet, als Betriebssystem kommt *Ubuntu 14.04 LTS Server 64 bit* mit Linux-Kernelversion 3.13 zum Einsatz. Die Netzwerkschnittstellen wurden mittels `ethtool` auf 100 Mbit/s konfiguriert. Auf dem Empfänger wurde mittels `netem` eine künstliche Verzögerung von 50 ms eingestellt. Der Testverkehr wird mit dem Programm `Iperf` (Client und Server, Version: 2.0.5) erzeugt, welches immer versucht mit maximal möglicher Datenrate zu senden. Im Linux-Kernel wurde die Option `tcp_no_metrics_save=1` gesetzt, womit ein Cache von verbindungsspezifischen Variablen deaktiviert wird, welche beim Wiederholen von Testläufen u. U. die Ergebnisse verfälschen kann. Zur Untersuchung des Sendeverhaltens der TCP-Verbindungen wurde der Netzwerkverkehr auf den beiden Sendern jeweils mittels `tcpdump` aufgezeichnet.



(a) TCP Reno *Durchsatz*

(b) TCP Vegas *Durchsatz*

(c) TCP Reno *RTT*

(d) TCP Vegas *RTT*

Abbildung 4: Zwei TCP Reno Sender und zwei TCP Vegas Sender getrennt über separate physische Pfade

B. Separate Pfade

Zur Überprüfung der Annahme, dass sich inkompatible TCP-Varianten auf separaten physischen Pfaden gegenseitig nicht negativ beeinflussen, wurde folgendes Experiment durchgeführt:

Sender 1 und Sender 2 nutzen jeweils eine TCP Reno- und eine TCP Vegas-Verbindung zur Übertragung von Daten an den Empfänger. Beide Sender nutzen jeweils eine der beiden Netzwerkschnittstellen für die TCP Reno-Verbindung und die andere Schnittstelle für die TCP Vegas-Verbindung. Auf dem Switch wurden statische Flows gesetzt, welche die beiden TCP Reno-Verbindungen auf die erste, und die beiden TCP Vegas-Verbindungen auf die zweite Netzwerkschnittstelle des Empfänger leiten.

Abb. 4 zeigt Durchsatz und RTT der vier Verbindungen. In Abb. 4a ist zu erkennen, wie die beiden TCP Reno-Verbindungen um die zur Verfügung stehende Übertragungskapazität auf ihrem Pfad konkurrieren. Dabei füllen sie die Warteschlange immer wieder bis zum Paketverlust, was an der erhöhten RTT in Abb. 4c zu erkennen ist. Die RTT der beiden TCP Vegas-Verbindungen (Abb. 4c) ist, nach der initialen *Slow Start*-Phase, gleichbleibend gering. Die stark gefüllte Warteschlange auf dem TCP Reno-Pfad hat somit keine negativen Auswirkungen auf die TCP Vegas-Verbindungen.

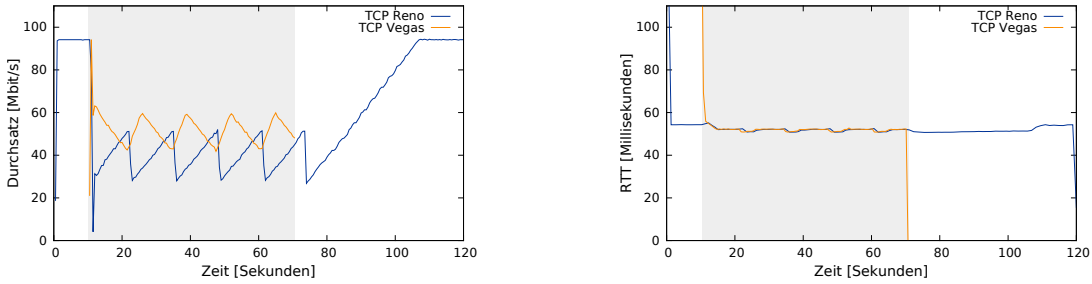
Wie Abb. 4b zeigt, wird die Übertragungskapazität des zweiten Pfades fair auf die beiden TCP Vegas-Verbindungen verteilt. Es ist zu erkennen, dass der Durchsatz der beiden TCP Vegas-Verbindungen nicht exakt gleich ist, dieses Verhalten ist für TCP Vegas jedoch nicht unüblich und kann auch in einfachen Experimenten zur Intraprotokollfairness von TCP Vegas beobachtet werden. Beim Vergleich von Abb. 4a und Abb. 4b ist gut zu erkennen, dass die Fairness zwischen den beiden TCP Vegas Sendern deutlich besser ist als zwischen den TCP Reno Sendern und dass TCP Vegas ein wesentlich gleichmäßigeres Sendeverhalten hat als TCP Reno.

Das Experiment hat gezeigt, dass TCP Vegas-Verbindungen, die in einem OpenFlow-Switch mittels statischer Flow-Regeln einen eigenen Pfad zugewiesen bekommen, nicht von TCP Reno-Verbindungen unterdrückt werden, die auf einem anderen Pfad weitergeleitet werden. Die Funktionsweise des Mechanismus hängt hierbei nicht von der Anzahl der OpenFlow-Switches, oder von der Tatsache, dass die Flow-Regeln statisch programmiert wurden, ab. Somit können die Ergebnisse auf den Einsatz des Mechanismus in realen SDN-Netzen, bei denen ein Controller die Flow-Regeln dynamisch programmiert, verallgemeinert werden.

C. OpenFlow-Meters *mittels App*

Zur Evaluation des *Meter*-Ansatzes und der in Abschnitt IV beschriebenen App, wurde ein weiteres Experiment durchgeführt. Anders als bei den getrennten physischen Pfaden wird auf jedem Server nur eine Netzwerkschnittstelle genutzt. Sender 1 überträgt Daten mittels TCP Reno zum Empfänger, Sender 2 mittels TCP Vegas. Die beiden Verbindungen konkurrieren somit auf dem Übertragungsabschnitt zwischen Switch und Empfänger miteinander. Die Flows und *Meter*-Begrenzungen werden dynamisch von der App gesetzt, wie in Abschnitt IV-B beschrieben.

Um das dynamische Verhalten der App beobachten zu können, startet Sender 1 (TCP Reno) zuerst und sendet



(a) Durchsatz (b) RTT

Abbildung 5: Durch den Einsatz eines OpenFlow-*Meters*, der über eine SDN-App dynamisch gesetzt wird, wird die TCP Vegas-Verbindung von der konkurrierenden TCP Reno-Verbindung nicht unterdrückt.

insgesamt für `120 s`. Sender 2 beginnt `10 s` später und sendet `60 s` lang.

Der Durchsatz beider Sender ist in Abb. 5a dargestellt, die Zeit in der Sender 2 aktiv ist, ist in der Abbildung grau unterlegt. Zunächst kann Sender 1 (TCP Reno) die volle Übertragungskapazität nutzen. Nach `10 s` kommt Sender 2 (TCP Vegas) hinzu und die App setzt die *Meter*-Begrenzung auf 50 Mbit/s. Daraufhin bricht der Durchsatz der TCP Reno-Verbindung stark ein und nähert sich dann wieder linear der Grenze von 50 Mbit/s an. Wird die Grenze überschritten, kommt es durch den Meter zu einem Paketverlust und TCP Reno halbiert das Staukontrollfenster. Dies wiederholt sich so lange periodisch, bis der *Meter* wieder deaktiviert wird. Danach steigt die Übertragungsrate linear an, bis die volle Übertragungskapazität des Senders erreicht ist.

Nach einem Paketverlust nutzt Sender 1 deutlich weniger Übertragungskapazität als ihm zusteht. Die TCP Vegas-Verbindung (von Sender 2) erkennt dies und nutzt die freie Kapazität ohne jedoch die koexistierende TCP Reno-Verbindung zu beeinträchtigen. Steigt die Rate von TCP Reno an, gibt TCP Vegas die entsprechende Kapazität frei. Sendet TCP Reno mit einer Übertragungsrate nahe der *Meter*-Begrenzung, fällt die Rate von TCP Vegas knapp unter die von TCP Reno.

TCP Reno wird durch den *Meter* daran gehindert die Warteschlange des Switches merklich zu füllen, was an der gleichbleibenden Umlaufzeit (Abb. 5b) zu erkennen ist. Es fällt auf, dass sich auch bei inaktivem *Meter*, also bevor die TCP Vegas-Verbindung aufgebaut ist und nach deren Ende, keine Warteschlangenverzögerung aufbaut. Dies liegt daran, dass Sender 1 der einzige Sender ist und nicht schneller senden kann, als der Switch die Daten weiterleitet.

Im Durchschnitt erreicht die TCP Vegas-Verbindung eine höhere Übertragungsrate als die TCP Reno-Verbindung. Dies hat zwei Gründe. Zum einen ist die TCP Vegas-Verbindung nicht durch einen Meter begrenzt und kann somit mehr als 50 % der Übertragungskapazität nutzen, sofern die TCP Reno-Verbindung ihren Anteil nicht voll ausschöpft. Zum anderen ist das Verhalten der Vegas-Staukontrolle auf eine geringe Pufferauslastung optimiert. Stellt sie einen drohenden Stau fest, reduziert sie ihre Senderate nur leicht aber dafür frühzeitig. Dies führt dazu dass die TCP Vegas-Verbindung die freie Kapazität sehr gut ausschöpfen kann ohne eine Überlastung des Netzes hervorzurufen. TCP Reno hingegen reagiert nur auf Paketverluste und geht dann von einer überfüllten Warteschlange aus. Daraufhin wird das Staukontrollfenster der Verbindung halbiert, was bei der kurzen Warteschlange auch ungefähr zu einer Halbierung der Senderate führt.

Das Verhalten von TCP Reno bei einer geringen Pufferauslastung unterscheidet sich somit von dem in Abschnitt II beschriebenen Verhalten, bei dem die TCP Reno-Verbindung den Puffer ungehindert füllen konnte (vgl. Abb. 1 in Abschnitt II). Dort führte der Abbau der Warteschlangenverzögerung nach einem Paketverlust zu einer merklichen Reduktion der RTT. Dadurch hatte die Halbierung des Staukontrollfensters wiederum einen deutlich geringeren Effekt auf die Senderate.

Das Experiment hat somit gezeigt, dass durch den Einsatz von OpenFlow-*Meters* verhindert werden kann, dass sich inkompatible TCP-Varianten gegenseitig unterdrücken. Eine TCP Reno-Verbindung kann die ihr zustehende Übertragungskapazität jedoch nur zu circa 75 % nutzen. Die ungenutzte Kapazität kommt der konkurrierenden TCP Vegas-Verbindung zugute.

Stehen in einem SDN-Netzwerk, zumindest abschnittsweise, keine separaten Pfade zur Verfügung, stellt der Einsatz von OpenFlow-Meters einen alternativen Mechanismus dar, um die Fairness beim Parallelbetrieb inkompatibler TCP-Varianten zu verbessern.

D. Begrenzung von TCP Vegas-Verbindungen

Wie das vorangegangene Experiment zeigt, wird die Senderate der TCP Reno-Verbindung nur durch den OpenFlow-*Meter*, nicht jedoch durch die konkurrierende TCP Vegas-Verbindung begrenzt. Es ist somit nicht nötig, die TCP Vegas-Verbindung ebenfalls durch einen *Meter* zu begrenzen um die Fairness zu verbessern. Wird eine TCP Vegas-Verbindung dennoch durch einen *Meter* begrenzt, wird das übliche Protokollverhalten gestört.

Abb. 6 zeigt den Durchsatz einer TCP Vegas-Verbindung von Sender 1 zum Empfänger. Der zugehörige Flow wurde durch einen *Meter* auf 50 Mbit/s begrenzt. Da sich bei Annäherung an diese Rate aber keine

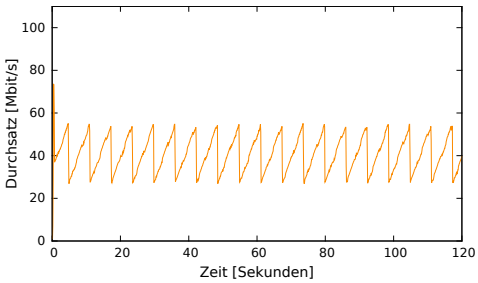


Abbildung 6: TCP Vegas begrenzt durch OpenFlow Meter

Warteschlange aufbaut, bekommt TCP Vegas kein frühes Stausignal und kann seine Senderate daher nicht proaktiv reduzieren. Stattdessen wird die Senderate solange erhöht, bis der *Meter* beginnt Pakete zu verwerfen. Die TCP Vegas-Verbindung fällt somit auf ein TCP Reno-artiges Verhalten zurück, was zu einer schlechten Ausnutzung der zur Verfügung stehenden Übertragungskapazität führt. Daher sollten OpenFlow-*Meter* nur zur Begrenzung von Verbindungen mit verlustbasierter Staukontrolle eingesetzt werden.

VI. Verwandte Arbeiten

Die hybride TCP-Variante *YeAH TCP* [11] versucht ebenfalls die Warteschlangenverzögerung gering zu halten, schützt sich aber eigenständig davor von aggressiveren Varianten unterdrückt zu werden. Eine YeAH TCP-Verbindung nutzt dazu eine Heuristik, mit der sie erkennt, ob sie gerade mit einer aggressiveren TCP-Variante konkurriert. Ist dies nicht der Fall, versucht sie die Warteschlangenverzögerung gering zu halten. Anderenfalls wechselt YeAH TCP in einen *Kompatibilitätsmodus* und verhält sich wie ein verlustbasiertes Verfahren. Das Ziel eine geringe Warteschlangenverzögerung zu verursachen wird somit gezielt aufgegeben, wenn die Gefahr besteht von einer anderen Verbindung unterdrückt zu werden.

Beim *Active Queue Management (AQM)* versucht das Netz für eine geringe Auslastung der Warteschlangen zu sorgen, indem die Router frühzeitig Stausignale an die Sender übermitteln. Hierzu kann *Explicit Congestion Notification (ECN)* (RFC 3168 [3]) zum Einsatz kommen, sofern dies von Sender und Empfänger unterstützt wird. Ist dies nicht der Fall, kann das AQM-Verfahren auch Pakete verwerfen, was vom Sender als implizites Stausignal interpretiert wird. Um die Fairness zwischen unterschiedlichen Verbindungen zu verbessern, können AQM-Verfahren mit speziellen *Scheduling-Strategien*, wie z. B. *Stochastic Fairness Queuing (SFQ)* [12] kombiniert werden (vgl. [13]). Ein solches kombiniertes Verfahrens ist *AFCD* ([14]). Am Beispiel der TCP-Varianten Reno und Cubic zeigt [14], dass damit auch die Fairness zwischen Verbindungen unterschiedlicher TCP-Varianten verbessert wird. Cubic ist, wie Reno, ebenfalls verlustbasiert. In den Experimenten mit verschiedenen AQM-Verfahren (u. a. RED [4] und CoDel [15]) hatte sich aber eine deutliche Unfairness zwischen den beiden Verbindungen gezeigt, die durch das AFCD-Verfahren wieder aufgehoben werden konnte. Im OpenFlow-Standard ist der Einsatz von AQM-Verfahren jedoch nicht vorgesehen.

Die Nutzung von separaten Warteschlangen für unterschiedliche Verkehrsklassen und der Einsatz von Ratelimitern ist nicht nur in SDN-Netzen möglich. In klassischen IP-Netzen stehen zwei Ansätze für die Umsetzung solcher QoS-Mechanismen zur Verfügung. Bei den *Differentiated Services (DiffServ)* (RFC 2474 [16]) werden IP-Pakete von den Routern am Netzrand klassifiziert und es wird im IP-Paket ein *DSCP-Codepoint* gesetzt, der von Routern im Netzinneren entsprechend interpretiert wird. Bei den *Integrated Services (IntServ)* RFC 2210 [17] können Endsysteme Netzressourcen mit Hilfe des *RSVP*-Protokolls reservieren. In einem gemischten Netz, bestehend aus OpenFlow-Switches und klassischen Routern, können sich die jeweiligen QoS-Mechanismen gegenseitig ergänzen. Die DSCP-Codepoints können auch von OpenFlow-Switches gesetzt und interpretiert werden. Eine Aushandlung über das *RSVP*-Protokoll könnte durch den SDN-Controller realisiert werden.

VII. Zusammenfassung und Ausblick

In dieser Arbeit wurde untersucht, welche Möglichkeiten in *OpenFlow*-basierten SDN-Netzen bestehen, um die Koexistenz inkompatibler TCP-Varianten zu ermöglichen. SDN bietet eine hohe Flexibilität bei der Wegewahl im Netz. Dies kann genutzt werden, um inkompatible Verbindungen auf separaten Pfaden durch das Netz zu leiten. Die Verbindungen sind somit effektiv voneinander getrennt und beeinflussen sich nicht gegenseitig. Allerdings ist dieser Ansatz nur möglich, wenn redundante Pfade im Netz zur Verfügung stehen.

Stehen keine redundanten Pfade zur Verfügung, können bei einigen OpenFlow-Switches stattdessen auch QoS-Mechanismen genutzt werden. Diese Arbeit konzentriert sich hierbei auf die Untersuchung von OpenFlow-*Meters*. Für den *OpenDaylight* SDN-Controller wurde in diesem Zusammenhang eine *App* entwickelt, die dynamisch Flows programmiert und bei Bedarf *Meter*-Begrenzungen setzt. Dies wurde in einem Testbett auf physischer Hardware evaluiert. Dabei hat sich gezeigt, dass verzögerungsbasierte TCP-Varianten mithilfe von OpenFlow-*Meters* vor der Unterdrückung durch verlustbasierte Varianten geschützt werden können und geringe Verzögerungen erreicht werden.

Bei den Experimenten hat sich weiterhin herausgestellt, dass eine *Meter*-Begrenzung nur für verlustbasierte Verbindungen sinnvoll ist. Werden verzögerungsbasierte Verbindungen durch einen *Meter* begrenzt, wird ihr Protokollverhalten gestört. Sie benötigen den Aufbau einer leichten Warteschlangenverzögerung als Stausignal. Der *Meter* verwirft jedoch Pakete bevor sich eine entsprechende Verzögerung aufbaut. Da sich verzögerungsbasierte Varianten bei einem drohenden Stau frühzeitig zurückziehen, ist es dennoch nicht notwendig diese durch einen *Meter* zu begrenzen.

Für verlustbasierte TCP-Varianten sind Paketverluste hingegen das reguläre Stausignal. Sie sind allerdings darauf ausgelegt, dass ein Paketverlust erst bei einer Überfüllung der Warteschlange auftritt. Da der *Meter* bereits Pakete verwirft bevor sich die Warteschlange füllt, führt dies zu einem etwas niedrigeren Durchsatz. Die nicht genutzte Kapazität kann jedoch stattdessen von einer konkurrierenden

verzögerungsbasierten Verbindung genutzt werden und ist somit nicht verloren.

Als weiteren QoS-Mechanismus sieht der OpenFlow-Standard, neben den *Meters*, die Nutzung separater Warteschlangen mittels der *setQueue-Action* vor. Hierbei sind ähnlich gute Ergebnisse zu erwarten wie bei der Nutzung separater Pfade. Darüber hinaus bieten separate Warteschlangen den Vorteil, dass ungenutzte Übertragungskapazität automatisch auf die restlichen Warteschlangen verteilt wird. Eine experimentelle Untersuchung dieses Ansatzes ist für eine Folgearbeit geplant.

Danksagung

Diese Arbeit wurde durch das bwNET100G+ Projekt des Ministeriums für Wissenschaft, Forschung und Kunst Baden-Württemberg (MWK) unterstützt. Die Autoren allein sind für den Inhalt der Arbeit verantwortlich.

Literatur

[1] J. Gettys and K. Nichols, "Bufferbloat: Dark buffers in the internet," *Commun. ACM*, vol. 55, no. 1, pp. 57–65, Jan. 2012. [Online]. Available: http://doi.acm.org/10.1145/2063176.2063196

[2] T. Flach, N. Dukkipati, A. Terzis, B. Raghavan, N. Cardwell, Y. Cheng, A. Jain, S. Hao, E. Katz-Bassett, and R. Govindan, "Reducing web latency: The virtue of gentle aggression," *SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 159–170, Aug. 2013. [Online]. Available: http://doi.acm.org/10.1145/2534169.2486014

[3] K. Ramakrishnan, S. Floyd, and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP," RFC 3168 (Proposed Standard), Internet Engineering Task Force, Sep. 2001, updated by RFCs 4301, 6040. [Online]. Available: http://www.ietf.org/rfc/rfc3168.txt

[4] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *Networking, IEEE/ACM Transactions on*, vol. 1, no. 4, pp. 397–413, Aug 1993.

[5] S. Floyd, "Congestion Control Principles," RFC 2914 (Best Current Practice), Internet Engineering Task Force, Sep. 2000, updated by RFC 7141. [Online]. Available: http://www.ietf.org/rfc/rfc2914.txt

[6] M. Allman, V. Paxson, and W. Stevens, "TCP Congestion Control," RFC 2581 (Proposed Standard), Internet Engineering Task Force, Apr. 1999, obsoleted by RFC 5681, updated by RFC 3390. [Online]. Available: http://www.ietf.org/rfc/rfc2581.txt

[7] L. S. Brakmo, S. W. O'Malley, and L. L. Peterson, "Tcp vegas: New techniques for congestion detection and avoidance," in *Proceedings of the Conference on Communications Architectures, Protocols and Applications*, ser. SIGCOMM '94. New York, NY, USA: ACM, 1994, pp. 24–35. [Online]. Available: http://doi.acm.org/10.1145/190314.190317

[8] K. Tan, J. Song, Q. Zhang, and M. Sridharan, "A compound tcp approach for high-speed and long distance networks," in *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, April 2006, pp. 1–12.

[9] P. Yang, W. Luo, L. Xu, J. Deogun, and Y. Lu, "Tcp congestion avoidance algorithm identification," in *Distributed Computing Systems (ICDCS), 2011 31st International Conference on*, June 2011, pp. 310–321.

[10] O. N. Foundation, "Openflow management and configuration protocol," Oktober 2013, https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow-config/of-config-1.2.pdf.

[11] A. Baiocchi, A. P. Castellani, and F. Vacirca, "Yeah-tcp: yet another highspeed tcp," in *Proc. PFLDnet*, vol. 7, 2007, pp. 37–42.

[12] P. McKenney, "Stochastic fairness queueing," in *INFOCOM '90, Ninth Annual Joint Conference of the IEEE Computer and Communication Societies. The Multiple Facets of Integration. Proceedings, IEEE*, Jun 1990, pp. 733–740 vol.2.

[13] F. Baker and R. Pan, "On Queuing, Marking, and Dropping," Internet-Draft, Internet Engineering Task Force, Work in Progress, May 2015, https://tools.ietf.org/html/draft-ietf-aqm-fq-implementation-02.

[14] L. Xue, S. Kumar, C. Cui, P. Kondikoppa, C.-H. Chiu, and S.-J. Park, "Afcd: An approximated-fair and controlled-delay queuing for high speed networks," in *Computer Communications and Networks (ICCCN), 2013 22nd International Conference on*, July 2013, pp. 1–7.

[15] K. Nichols, V. Jacobson, A. McGregor, and J. Iyengar, "Controlled Delay Active Queue Management," Internet-Draft, Internet Engineering Task Force, Work in Progress, April 2015, https://tools.ietf.org/html/draft-ietf-aqm-codel-01.

[16] K. Nichols, S. Blake, F. Baker, and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers," RFC 2474 (Proposed Standard), Internet Engineering Task Force, Dec. 1998, updated by RFCs 3168, 3260. [Online]. Available: http://www.ietf.org/rfc/rfc2474.txt

[17] J. Wroclawski, "The Use of RSVP with IETF Integrated Services," RFC 2210 (Proposed Standard), Internet Engineering Task Force, Sep. 1997. [Online]. Available: http://www.ietf.org/rfc/rfc2210.txt

Radio Resource Allocation in LTE Femtocell Considering QoE

Ming Li, Phuong Nga Tran and Andreas Timm-Giel
Communication Networks, Hamburg University of Technology
Schwarzenbergstraße 95E, 21073 Hamburg, Germany
{ming.li, phuong.tran, timm-giel}@tuhh.de

***Abstract*—In this paper, we study the QoE-based resource allocation problem for the downlink direction in LTE femtocells, considering the transport network limitation. The problem is first formulated as a convex optimization problem, which maximizes the aggregated QoE of all users and is then solved by Lagrangian relaxation method. This approach provides an optimal resource allocation solution but it is not practical, because the allocated bandwidth in the model is assumed to be a real value, while in practice, only an integer amount of Physical Resource Blocks (PRBs) can be assigned to a user. This integer constraint makes the problem non-convex and very hard to solve in real time. We therefore propose an efficient heuristic algorithm, which can be applied in LTE systems and take the optimal solution obtained from the Lagrangian method as the benchmark for our algorithm. The heuristic is mathematically proven to provide an optimal solution in case there is no limitation in the transport network. In case of transport network limitation, the simulation results show that it gives almost identical performance (ca. 1.5%) as the optimal solution in the investigated scenario.**

Keywords— LTE femtocell; QoE; radio resource allocation; utility; Lagrangian relaxation

I. INTRODUCTION

In mobile communications, it is an acknowledged fact that most of the mobile data is generated indoors. Subscribers use mobile data services 25% of the time inside their offices, 40% from inside their homes, and 35% on the move in public areas with at least 80% of the traffic coming from indoor locations [1]. Usually the conventional solution for increasing network coverage is adding more network infrastructure such as tower cells or transmitters. However, the deployment of these macrocell elements can be very costly and increase the inter-cell interference in the network due to limited spectrum resources. With increasing number of indoor mobile users, operators have to find a better cost efficient solution to provide necessary Quality of Service (QoS) to the users. The latest trend to solve this problem is to introduce a large number of femtocells, which can be connected to the operator's network by using existing user broadband (e.g. xDSL / Cable / Fiber) connection, for indoor coverage. With this approach, operators can save costs for deploying more macrocells and offload some traffic from macrocells to femtocells. This can lead to a decrease in capital expenditures (CAPEX) and operating expenses (OPEX) of operators and makes femtocell deployment a lucrative option [2]. An increase in user satisfaction is also expected due to possible better indoor reception and longer battery life because of low transmission power levels [3].

However, deploying femtocells raises many new challenges, one of which is the resource allocation for users served by femtocells. Different from macrocells, whose transport network is usually appropriately dimensioned, the one of femtocells is limited by the existing user's broadband internet connection (e.g. xDSL). Therefore, the resource allocation must not only consider the radio resources constraints but also the transport network limitations.

Nevertheless, resource allocation in either macrocells or femtocells has the same objectives to improve the network performance and the Quality of Experience (QoE), which represents the users' satisfaction level. The resource allocation approaches focusing on network performance usually consider the fairness, latency reduction, spectral efficiency and system utilization as the main objective. The approaches focusing on QoE often aim to maximize the aggregated QoE, which can be represented by a utility function.

The QoE is not a linear function of QoS parameters and very much depends on the application. For example, for a user streaming high quality audio the difference in terms of user satisfaction when served with 2Mbit/s instead of 1 Mbit/s is low, if not negligible, whereas the user is obviously much more content when served with 100 instead of 50 Kbit/s. Hence, QoE is a more important indicator to measure the network quality provided by operators from the users' point of view. For that reason, in this paper, we will study the problem of resource allocation in LTE femtocells considering the transport network limitations and focusing on improving QoE.

II. STATE OF THE ART

Utility-based radio resource allocation has been widely researched. A utility-based adaptive bandwidth sharing for elastic traffic is proposed in [4] for wired networks. This is formulated as a concave maximization problem with linear constraints. It is shown by the Karush–Kuhn–Tucker (KKT) conditions [4] that the optimal bandwidth allocation can be achieved. The utility-based optimal resource allocation is researched in wireless networks in [5], and [6] considering the channel conditions. In [5], the authors further prove that the performance difference in their proposed mechanism and the optimal solution is bounded. A unified utility function for QoS traffic and best effort traffic is proposed in [6]. [7] uses utility functions based on the Exponential Moving Average (EMA) rate achieving better fairness. In [8], the Mean Opinion Score (MOS) is used as a common utility metric for the quality of experience and the performance and feasibility are studied in LTE networks. However, all the above-

mentioned papers do not consider the transport limitations, since it is often well dimensioned and hence is not a problem for macrocells. To our best knowledge, only two papers considered the limited transport network in the scheduling mechanism so far [9] [10]. Nevertheless, in [9], a simple fair rate scheduler is proposed without considering the QoE. [10] proposes a QoE based dynamic transport architecture adapted to video service requirements on the conceptual level. Therefore, it motivates us to study the resource allocation problem for LTE femtocells considering transport network limitations. The main contributions of this work are listed as follows:

1. An analytical model based on convex optimization to estimate the upper bound of the achievable aggregated QoE.
2. An efficient heuristic algorithm, which is proven to be optimal in case of no transport limitation. This heuristic is therefore also applicable to LTE macrocells system. If the transport network is limited as it is often the case in LTE femtocells, the simulation has shown that the heuristic gives a close to optimal solution.

The rest of this paper is organized as follows. Section III presents the problem formulation and introduces the solution algorithms. In Section IV, the LTE simulation model is introduced and the performance of the proposed algorithm is studied using simulations. Section V summarizes and concludes the paper.

III. Problem Formulation and Solution Algorithm

A. QoE Functions

QoE is often measured by Mean Opinion Score (MOS), which ranges from 1 to 5, with 1 being the worst and 5 being the best experience. Fig. 1 is an example of QoE functions for different applications [11], [12].

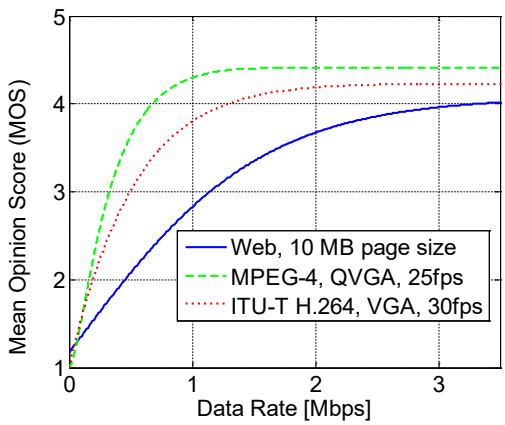


Fig. 1. MOS over data rate

The blue curve shows the MOS over data rate for the web user based on the ITU-T G.1030 [11] assuming the web has a 60 seconds duration context with a 10MB page size. The red and green curves show the MOS behavior for the video user with transcoding based on ITU-T G.1070 [12]. It reveals that the MOS value representing QoE monotonously increases with the data rate, while the marginal MOS value decreases with the data rate. It is because doubling the data rate for example does not double the users' satisfaction. For example, if a user is already quite satisfied with 1 Mbps data rate, doubling the data rate to 2 Mbps cannot double the user satisfaction. For the Best Effort traffics and video with transcoding, the MOS (QoE) function is hence a strictly concave function.

In this paper, we use the concave part of a sigmoid function shown in (1) as the QoE function. The sigmoid function is very convenient to model the MOS due to its convergence, since MOS is bounded to 5.

$$u(r) = \frac{A}{1+e^{-\alpha r}} + D \tag{1}$$

where the utility $u(r)$ is represented as a function of data rate r. The parameters α, A and D control the shape of the curve. By curve fitting method, QoE functions for different applications can be constructed.

B. System Model and Assumptions

In this paper, we consider a network scenario of one femtocell serving multiple mobile users. The channel quality changes over time due to the mobility, fast fading and the interference from macrocells.

In LTE, the base station (eNB) schedules its available bandwidth resources in terms of PRBs (Physical Resource Blocks) to the users periodically every TTI (Transmission Time Interval, 1ms). A PRB which has 180 kHz bandwidth is the minimum resource unit which can be allocated to a user in LTE. The allocated number of PRBs and the MCS (Modulation and Coding Scheme) value of a user are used to look up the Transport Block Size (TBS). The TBS is the amount of data that the user can transmit in the current TTI. From this, the achievable throughput over the air interface is determined, as illustrated in Fig. 2 [17]. For instance, the MCS 4 means QPSK (Quadrature Phase Shift Keying) modulation scheme with a coding rate of 0.6.

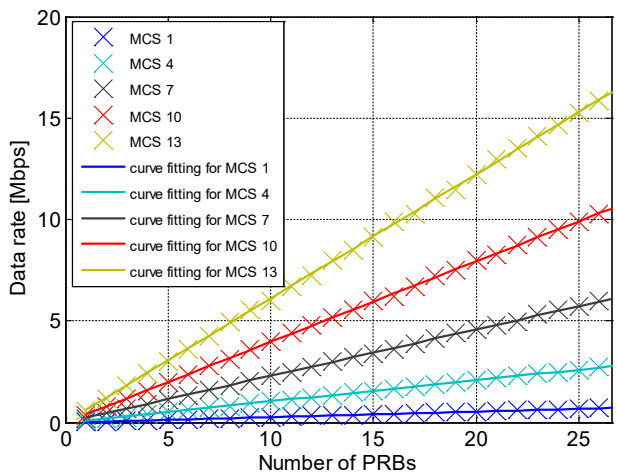


Fig. 2. LTE Throughput over Number of PRBs

It can be seen from the figure, that the data rate has a near linear relationship with the number of PRBs, which identifies the allocated bandwidth to a user. Therefore, without significant loss of accuracy, the data rate r_i of user i is approximately proportional to the allocated bandwidth b_i, which can be described by the linear function $r_i = b_i \cdot \sigma_i$. The coefficient σ_i is considered as the channel quality indicator and varies with the MSC. It is pre-computed for each MSC based on the data provided by 3GPP.

C. Problem Formulation and Solution

The resource allocation problem at each TTI is formulated as follows:

$$\begin{aligned} &\max U, \quad U = \sum_i u_i(r_i) = \sum_i \left(\frac{A}{1+e^{-\alpha_i \cdot r_i}} + D \right) \\ &s.t. \quad \sum_i \frac{r_i}{\sigma_i} \le B_c; \quad \sum_i r_i \le R_{S1} \\ &\qquad \forall i \quad r_i \ge 0 \end{aligned} \tag{2}$$

The objective is to maximize the aggregated QoE represented by a strictly concave QoE function U of the data rate r. The total bandwidth allocated to users is limited by the cell throughput B_c and the total achievable data rate under channel quality is limited by the transport network (S1) R_{S1}. It has been proven that the Hessian matrix is positive definite, so the optimization problem is convex. Besides, the Slater's condition is fulfilled, and therefore the convex problem has a strong duality. Therefore, we will apply the Lagrangian relaxation method to solve the problem. The proofs are omitted due to space limitation in this paper.

By introducing two non-negative dual variables and absorbing the two constraints into the optimization problem, the Lagrangian dual problem is shown as:

$$\min_{\{\lambda \ge 0\}} \left\{ \overbrace{\max_{\{r \ge 0\}} \left\{ \overbrace{\sum_i \left(\frac{A}{1+e^{-\alpha_i \cdot r_i}} + D \right) - \lambda_0 \left(\sum_i r_i - R_{S1} \right) - \lambda_1 \left(\sum_i \frac{r_i}{\sigma_i} - B_c \right)}^{L} \right\}}^{f} \right\}$$

The Lagrangian dual function $f = \max_{\{r \ge 0\}} \{L\}$ is reformulated and shown in formula (3). The utility function can be reformulated and expressed as a function of hyperbolic tangent function with the purpose of simplifying the mathematical calculation, e.g. the differentiation operation.

$$\begin{aligned} f &= \max_{\{r \ge 0\}} \left\{ \sum_i \left(\overbrace{\frac{A}{1+e^{-\alpha_i \cdot r_i}} + D - r_i \left(\frac{\lambda_1}{\sigma_i} + \lambda_0 \right)}^{L_i} \right) + \lambda_1 \cdot B_c + \lambda_0 \cdot R_{S1} \right\} \\ &= \max_{\{r \ge 0\}} \left\{ \sum_i \left(\overbrace{\frac{A}{2} \left[\tanh\left(\frac{\alpha_i \cdot r_i}{2} \right) + 1 \right] + D - r_i \left(\frac{\lambda_1}{\sigma_i} + \lambda_0 \right)}^{L_i} \right) + \lambda_1 \cdot B_c + \lambda_0 \cdot R_{S1} \right\} \\ &= \max_{\{r \ge 0\}} \left\{ \sum_i L_i + \lambda_1 \cdot B_c + \lambda_0 \cdot R_{S1} \right\} \end{aligned} \tag{3}$$

Lemma 1: $\forall i, r_i \ge 0, L_i$ is a concave function and has one and only one maximum L_i^*. Then the dual problem becomes:

$$\min_{\lambda} f = \min_{\lambda} \left\{ \sum_i L_i^* + \lambda_1 \cdot B_c + \lambda_0 \cdot R_{S1} \right\} \tag{4}$$

Proof: the function L_i is twice differentiable and its second derivative given below is negative with $r_i \ge 0$.

$$\begin{aligned} \frac{dL_i}{dr_i} &= \frac{A \cdot \alpha_i}{4} \left[1 - \tanh^2 \left(\frac{\alpha_i \cdot r_i}{2} \right) \right] - \left(\frac{\lambda_1}{\sigma_i} + \lambda_0 \right) \\ \frac{d^2 L_i}{dr_i^2} &= -\frac{A \cdot \alpha_i^2}{4} \tanh\left(\frac{\alpha_i \cdot r_i}{2} \right) \cdot \left[1 - \tanh^2 \left(\frac{\alpha_i \cdot r_i}{2} \right) \right] < 0 \end{aligned} \tag{5}$$

L_i reaches its maximum at r_i^* if $\frac{\partial L_i}{\partial r_i}(r_i^*) = 0$ and $r_i^* \ge 0$. Since L_i is a concave function with $r_i \ge 0$, if $\left. \frac{dL_i}{dr_i} \right|_{r_i = 0} < 0$, L_i is a monotonically deceasing function. Therefore, it reaches its maximum when $r_i = 0$. The formulation of L_i^* is as follows:

$$L_i^* = \max_{r_i \ge 0} L_i = \begin{cases} \frac{A}{2} \left[\tanh\left(\frac{\alpha_i \cdot r_i^*}{2} \right) + 1 \right] + D - r_i^* \left(\frac{\lambda_1}{\sigma_i} + \lambda_0 \right) & \frac{\lambda_1}{\sigma_i} + \lambda_0 \le \frac{A \cdot \alpha_i}{4} \\ \frac{A}{2} + D & \frac{\lambda_1}{\sigma_i} + \lambda_0 > \frac{A \cdot \alpha_i}{4} \end{cases} \tag{6}$$

with

$$r_i^* = \begin{cases} \frac{2}{\alpha_i} \operatorname{arctanh} \sqrt{1 - \frac{4\left(\frac{\lambda_1}{\sigma_i} + \lambda_0 \right)}{A \cdot \alpha_i}} & \frac{\lambda_1}{\sigma_i} + \lambda_0 \le \frac{A\alpha_i}{4} \\ 0 & \frac{\lambda_1}{\sigma_i} + \lambda_0 > \frac{A\alpha_i}{4} \end{cases} \tag{7}$$

Therefore, for $r_i \ge 0$, L_i has one and only one maximum L_i^*.

D. Subgradient Projection Method

The Lagrangian dual problem is now expressed as (8). It can be solved by a standard subgradient projection method to get the minimum of f. Due to the strong duality, f has one and only one minimum which is also the solution of the primal problem.

$$\begin{aligned} &\min_{\lambda} f, f = \sum_i L_i^* + \lambda_1 \cdot B_c + \lambda_0 \cdot R_{S1} \\ &s.t. \boldsymbol{\lambda} \ge 0 \end{aligned} \tag{8}$$

The subgradient projection method is an iterative method that starts with some initial feasible vector $\boldsymbol{\lambda}^0 \ge 0$, and generates the next iterate by taking a step along the negative subgradient direction $-s_k$ of f at λ^k and then, by projecting on the set $\boldsymbol{\lambda} \ge 0$ to maintain feasibility. The subgradients over the two dual variables are:

$$\mathbf{s}=\begin{bmatrix} s_0 \\ s_1 \end{bmatrix}=\begin{bmatrix} \dfrac{\partial f}{\partial \lambda_0} \\ \dfrac{\partial f}{\partial \lambda_1} \end{bmatrix}=\begin{bmatrix} R_{S1}+\sum_i \left(\dfrac{\partial L_i^*}{\partial \lambda_0}\right) \\ B_c+\sum_i \left(\dfrac{\partial L_i^*}{\partial \lambda_1}\right) \end{bmatrix}$$

with

$$\frac{\partial L_i^*}{\partial \lambda_0}=\begin{cases} -\dfrac{2}{\alpha_i}\operatorname{arctanh}\sqrt{1-\dfrac{4\left(\dfrac{\lambda_1}{\sigma_i}+\lambda_0\right)}{A\cdot\alpha_i}} & \dfrac{\lambda_1}{\sigma_i}+\lambda_0 \le \dfrac{A\alpha_i}{4} \\ 0 & \dfrac{\lambda_1}{\sigma_i}+\lambda_0 > \dfrac{A\alpha_i}{4} \end{cases}$$

$$\frac{\partial L_i^*}{\partial \lambda_1}=\begin{cases} -\dfrac{2}{\alpha_i\cdot\sigma_i}\operatorname{arctanh}\sqrt{1-\dfrac{4\left(\dfrac{\lambda_1}{\sigma_i}+\lambda_0\right)}{A\cdot\alpha_i}} & \dfrac{\lambda_1}{\sigma_i}+\lambda_0 \le \dfrac{A\alpha_i}{4} \\ 0 & \dfrac{\lambda_1}{\sigma_i}+\lambda_0 > \dfrac{A\alpha_i}{4} \end{cases}$$

Formally, a typical iteration of the subgradient projection method is given by $\boldsymbol{\lambda}^{k+1}=\left[\boldsymbol{\lambda}^k - t_k\cdot\mathbf{s}^k\right]^+$ with t_k being a positive step size at the iteration *k*. In this paper, we apply the Modified Polyak's step size:

$$t_k=\gamma\frac{f\left(\lambda^k\right)-\hat{f}_k}{\|s_k\|^2} \text{ with } \hat{f}_k=\min_{0\le j\le k} f\left(\lambda^k\right)-\delta$$

$$\rightarrow t_k=\gamma\frac{f\left(\lambda_k\right)-\min_{0\le j\le k} f\left(\lambda_k\right)+\delta}{\|\mathbf{s}^k\|^2}$$

γ,δ are positive parameters. $\|\mathbf{s}^k\|$ is the Euclidean distance at *k* iteration and it is calculated with $\|\mathbf{s}\|^2=s_0^{\ 2}+s_1^{\ 2}$. The iterations are stopped when $f\left(\boldsymbol{\lambda}_k\right)-\min_{0\le j\le k} f\left(\boldsymbol{\lambda}_k\right)\le\varepsilon$ (ε: a predefined value regarding to accuracy) for some iterations.

By solving the optimization problem, the optimal bandwidth allocated to all the active users are obtained, e.g. the bandwidth of user *i* is $b_i^*=\dfrac{r_i^*}{\sigma_i}$.

In practice, we can allocate only an integer amount of resource blocks (PRB) to a user and therefore the resulting allocated bandwidth b_i is not exactly the same as the one obtained by solving the optimization problem above. If the allocated PRB, which is an integer variable, is modeled in the formulation above, the optimization problem will become NP-hard [18]. Consequently, there is no polynomial-time algorithm to solve the problem.

Nevertheless, this solution gives an upper bound on the achievable total QoE, which can be considered as a benchmark for heuristic resource allocation algorithms.

E. Proposed heuristic

The proposed heuristic algorithm is shown in Fig. 3. Due to concavity, the marginal QoE is monotonically decreasing. The QoE gain of a user getting an additional PRB is always decreasing. As a result, the available PRBs are allocated to the user in a simple iterative manner. In each iteration, one available PRB is assigned to the user who has the maximum marginal QoE, and the marginal QoE of that user is updated for the next iteration. This heuristic is proven to result in the optimal solution for the optimization problem (1), in case the transport network is not the bottleneck, which is equivalent to removing the constraint $\sum_i r_i=\sum_i b_i\cdot\sigma_i\le R_{S1}$ from the formulation (see Appendix). To be noticed, 0.18 represents the spectrum bandwidth of a PRB which is 180 Khz.

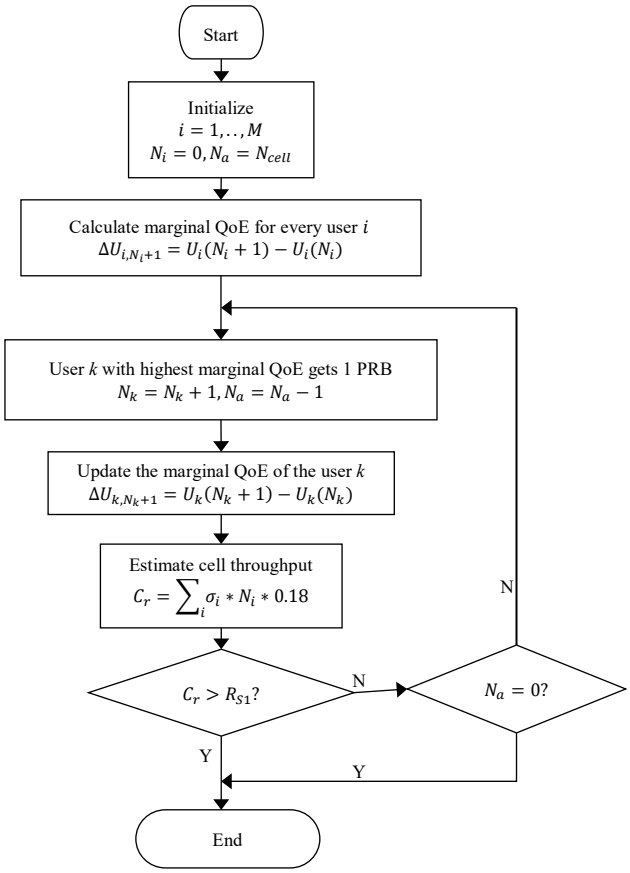


Fig. 3. The flowchart of the proposed heuristic

TABLE I. NOTATIONS

i	User index
M	Number of users
N_i	Number of PRBs allocated to user *i*
N_a	Total available PRBs in the cell after HARQ (Hybrid Automatic Repeat Request)
N_{cell}	Number of PRBs in the cell
$\Delta U_{i,n}$	Marginal QoE of user *i* with *n* PRBs
U_i	QoE of user i
C_r	Cell throughput in Mbps
σ_i	Sigma value of user i (channel condition)
R_{S1}	Transport network capacity in Mbps

However, in case of transport network limitations, it is important to stop allocating PRBs to users if estimated achievable cell throughput becomes larger than the transport network capacity. Otherwise, the transport link will be overloaded, and if so, the TCP flows tend to have equal share of the congested link due to TCP congestion control.

IV. System Model and Simulation Results

A. Simulation Model and Scenario

Our LTE simulator [13], [14] is implemented using the Riverbed modeler software (previously known as OPNET modeler, version 18) [15]. It includes all basic E-UTRAN (Evolved Universal Terrestrial Access Network) and EPC (Evolved Packet Core) network entities. Fig. 4 shows a scenario with 7 eNBs (1 tier cellular network), and a number of IP routers connecting the eNBs and the aGW (access gateway) in the E-UTRAN network. The user-plane protocols are implemented according to the 3GPP specifications. The EPC network entities are represented by the aGW network entity.

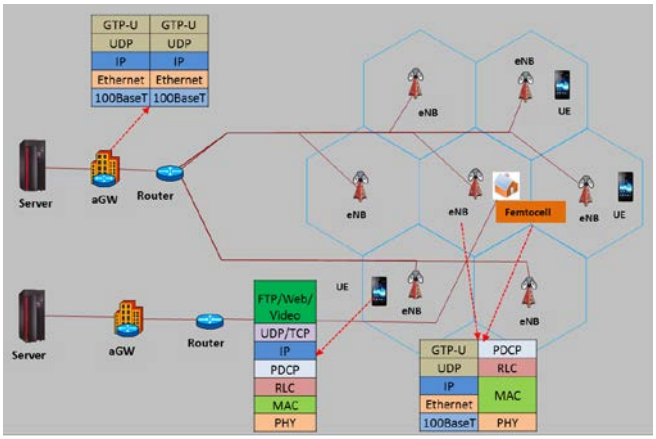


Fig. 4. LTE Simulation Model

The radio channel in the femtocell is modeled differently than in the macrocells. For the femtocell the indoor channel model is used and the femtocell transmission power is much lower. It is assumed that the femtocell is located inside a house. It is further assumed that the macrocells and the femtocells use the same frequency band, and the interference from macro eNBs to femtocells is modeled taken the penetration loss through the wall into consideration. Besides, the small scale frequency selective fading is modeled. The Exponential Effective SINR Mapping (EESM) is used for calculating the average channel condition which determines the MCS and TBS. In addition, an 8 stop-and-wait HARQ is modeled through a statistical method.

In order to evaluate our proposed algorithm, the corresponding radio resource allocation algorithms presented in Section III are implemented. The following investigations are based on the simulation settings given in TABLE II.

TABLE II. Simulation Settings

Parameter	Settings
Macro eNBs (cells) settings (fully loaded)	7 eNBs with hexagonal coverage, 500ms inter-eNB distance (center eNB located at the original point (0m, 0m)) Path loss: 128.1 + 37.6log10(R), R in Km [16] Small scale fading: 3GPP Pedestrian A Transmission power: 26 dBm per PRB
Femtocell settings	House size 20mx20m with the femtocell station in the middle with coordinate: (210m,0m) Penetration loss (interference from macro eNBs) over the wall: 12 dB mean with 8 dB std. Path loss: 41.1 + 16.9*log10(R), R in Km [16] Small scale fading: 3GPP Pedestrian A Transmission power: 0 dBm per PRB
Number of UEs in femtocell	4 Video users, 2 Web user, 2 FTP user
TCP version	New Reno with 64 kbytes receiver buffer size
Traffic type	Video: full buffer, TCP based (non-GBR) HTTP: 2 MB page size, reading time is 0s (non-GBR) FTP: 10 MB page size, idle (reading) time is 0s (non-GBR)
Mobility model	5 km/h, Random Direction
Transport limitation (aGW->femtocell)	16 Mbps, FIFO queue, 1 Mbyte buffer size
Scenario	50 PRBs (10 MHz)
Simulation time	1000s (5 runs with different seeds) with warm up period of 150s

B. QoE Functions

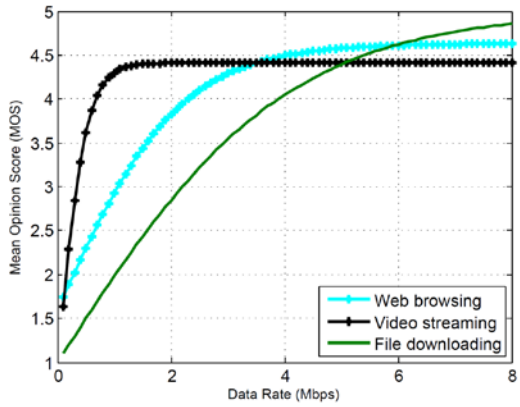


Fig. 5. QoE functions used in the simulation

Theoretically, the proposed approaches are applicable to any kind of concave functions. The sigmoid function is very convenient to model the MOS due to its convergence, since the MOS is bounded to 5. The video and the web curves shown in Fig. 1 based on the ITU-T reports [11], [12] are considered. By curve fitting, the parameters for different traffic types can be found which are shown in TABLE III. The QoE functions are plotted in Fig. 5.

TABLE III. Parameters for QoE Functions

	A	D	α
Video	6.954	-2.542	4.104
Web	5.815	-1.735	1.294
FTP	7	-2.5	0.5

C. Simulation Results

In this part, the performance of the proposed heuristic is evaluated by simulations and compared to the solution of the optimization problem. Besides, the heuristic without considering the transport limitation (marked as w/o S1) is added for comparison. It is been proven that without transport limitation, the heuristic is optimal (see Appendix). As a result, it is meaningful to check the performance in case of transport limitation.

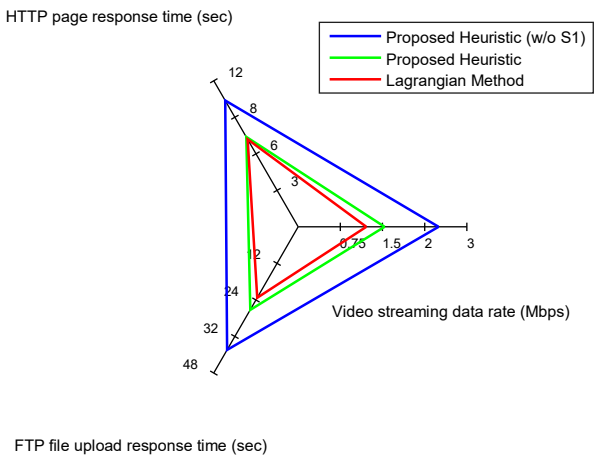


Fig. 6. Application performance comparison

Fig. 6 and Fig. 7 show the application performance and their corresponding MOS (the error bars show the 95% confidence interval). Fig. 7 shows the average user QoE calculated over multiple samples. For each sample, the MOS is calculated based on a 5 seconds batch mean of the throughput since the human beings have the sense of user satisfaction in seconds instead of milliseconds. The error bars show the 95% confidence intervals, but they are very small and hardly visible. Considering the minimum MOS value (scaling between 1 and 5) of each user is 1, the gain is calculated based on formula 7. MOS_2 and MOS_1 represents the aggregated MOS for all the users with arbitrary scheme 2 and 1, and N represents the total number of users.

$$G = \frac{(MOS_2 - N\cdot 1) - (MOS_1 - N\cdot 1)}{(MOS_1 - N\cdot 1)}\cdot 100\% = \frac{MOS_2 - MOS_1}{(MOS_1 - N)}\cdot 100\% \qquad (9)$$

For all three types of users, the application performance applying the heuristic approach and the optimal approach are quite close. Looking at the video streaming users, using the Lagrangian method and the proposed heuristic, the video streaming data rates are reduced significantly comparing to the heuristic without the awareness of the transport limitation. However, the MOS of video streaming users nearly remain the same as seen from Fig. 7. The reason is that the video MOS has non-linear relation with the data rates. Thus, the video users can remain satisfied even with less amount of resources. For example, the video streaming data rate is reduced from 2.08 Mbps with the heuristic without the awareness of the transport limitation to 1.01 Mbps with Lagrangian method, while the corresponding's MOS is only reduced from 4.41 to 4.3. For the heuristic without the awareness of the transport limitation, it is been observed that it allocates too much air interface resources to the users which put too much traffic to the transport network and cause the congestion. Due to the TCP congestion control, the TCP flows tend to have the equal share over the congested link. Consequently, video users have MOS around 4.4 while the FTP user has a MOS around 2.7.

On the contrary, the HTTP and FTP users get more resources and the download response times are reduced dramatically for the Lagrangian method and proposed heuristic. With little performance degradation of video users, the satisfactions of HTTP and FTP users are improved significantly. For instance, the FTP file download time is reduced from 42.1 seconds with the heuristic without the awareness of the transport limitation to 28.8 seconds with proposed heuristic and 24.2 seconds with Lagrangian method, while the corresponding's MOS is increased from 2.77 to 3.43 and 3.71 separately. Regarding to the aggregated QoE, the proposed heuristic achieves 32.2 while the Lagrangian method gives 32.6 and the gain is around 1.5% based on formula 7. The gain is around 11.1% for the Lagrangian method against the proposed heuristic without the awareness of the transport limitation.

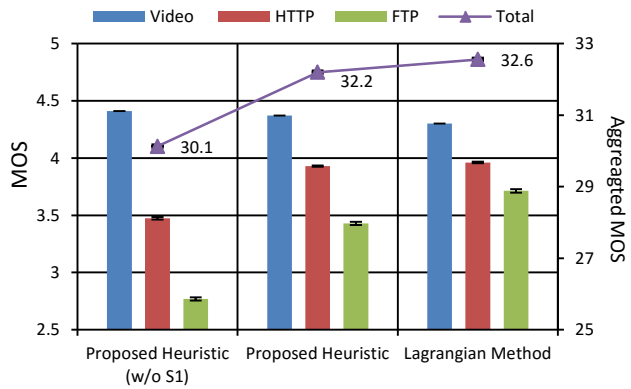


Fig. 7. MOS comparison

V. Conclusion and Future Work

The paper proposed a new approach to solve the QoE-based resource allocation problem in LTE femtocells. The innovation of this research is to consider the transport network limitations in radio resource allocation. We modeled the problem using convex optimization and solved it with Lagrangian relaxation method. This analytical solution was considered as the benchmark for our newly designed heuristic. Both analytical solution and heuristic algorithm were implemented in the LTE simulation model with three different traffic types, i.e. video, web and FTP. The simulation results showed that the heuristic provides a near-optimal solution. Furthermore, it provides a better QoE for FTP users compared to the conventional resource allocation approach, in which the transport network is not taken into account. Finally, we also proved mathematically that the heuristic provides an optimal solution when there is no transport network limitation.

In this paper, the fairness has not been considered and some users might get starved for a long time. We will cover this issue by taking the moving average rates into consideration in the future work. Besides, the extension to real-time delay sensitive traffic will be studied.

REFERENCES

[1] Cisco, "Cisco visual networking index : Global mobile data traffic forecast update , 2010-2015," *Growth Lakeland*, vol. 2011, no. 4, pp. 2010–2015.

[2] Z. Zhong, J. Hamalainen, and Y. Yang, "Practical resource scheduling and power control optimization for LTE femtocell networks," *Multi-Carrier Systems & Solutions (MC-SS), 2011 8th International Workshop on*, pp.1-5, 3-4 May 2011.

[3] Q. Zhang, X. Zhu, L. Wu, and K. Sandrasegaran, "A coloring-based resource allocation for OFDMA femtocell networks," *Wireless Communications and Networking Conference (WCNC), 2013 IEEE*, pp.673-678, 7-10 April 2013.

[4] A. Kumar, D. Manjunath, and J. Kuri, *Communication Networking: An Analytical Approach*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. 2004

[5] W.H. Kuo, and W. Liao, "Utility-based radio resource allocation for QoS traffic in wireless networks," *Wireless Communications, IEEE Transactions*, July 2008

[6] L. Chen, B. Wang, X. Chen, X. Zhang, and D. Yang, "Utility-based resource allocation for mixed traffic in wireless networks," *Computer Communications Workshops (INFOCOM WKSHPS), 2011 IEEE Conference*, April 2011

[7] M. Li, P. N. Tran, D. Wang, and A. Timm-Giel, "Radio resource allocation in LTE using utility functions based on Moving Average Rates, " IEEE WCNC'14 Track 2, April, 2014

[8] T. Zinner, T. Hoßfeld, T.N. Minhas, and M. Fiedler, "Controlled vs. uncontrolled degradations of QoE – the provisioning-delivery hysteresis in case of video," EuroITV 2010 Workshop: Quality of Experience for Multimedia Content Sharing, Tampere, 2010.

[9] H. Hariyanto, K.W. Noviyanti, A.K. Widiawan, and A. K. Hendrawan, "Backhaul-aware scheduling for WiMAX femtocell with limited backhaul capacity," *TENCON 2011 - 2011 IEEE Region 10 Conference* , Nov. 2011

[10] N. Amram, F. Bo, G. Kunzmann, T. Melia, D. Munaretto, S. Randriamasy, B. Sayadi, J. Widmer, and M. Zorzi, "QoE-based transport optimization for video delivery over next generation cellular networks," *ISCC, 2011 IEEE Symposium on*, Jun. 2011

[11] *Estimating end-to-end performance in IP networks for data applications*, ITU-T Recommendation G.1030, 2005.

[12] *Opinion model for video-telephony applications*, ITU-T Recommendation G.1070, 2012.

[13] N. Zahariev, Y. Zaki, T. Weerawardane, X. Li, C. Görg, and A. Timm-Giel. "Optimized Service Aware LTE MAC Scheduler with Comparison Against other Well Known Schedulers," WWIC 2012, Santorini, Greece, June 2012.

[14] M. Li, L. Zhao, Y. Zaki, A. Timm-Giel, and C. Görg. "Investigation of network virtualization and load balancing techniques in LTE networks," *Vehicular Technology Conference (VTC Spring), 2012 IEEE 75th*, Yokohama, Japan, May. 2012.

[15] Riverbed official website, http://www.riverbed.com/

[16] *Guidelines for evaluation of radio interface technologies for IMT-Advanced*, ITU-T Recommendation M.2135-1, 2009.

[17] 3GPP Technical Report TS 36.213. E-UTRA - Physical layer procedures. Technical Report version 9.3.0, 3rd Generation Partnership Project, September 2010.

[18] P. Manyem, and J. Ugon. "Computational Complexity, NP Completeness and Optimization Duality: A Survey," Electronic Colloquium on Computational Complexity, Report No. 9, 2012

APPENDIX

In case of no transport limitation, the problem (equ. 2) is equivalent to the following problem:

$$\max\left(\sum_{i=1}^{N} U_i(r_i)\right) \quad st. \quad \sum_{i=1}^{N} r_i = R \qquad r_i \in Z^+ \qquad \text{(Problem I)}$$

with N is the number of users, R is the number of resource blocks, r_i is the variable representing the number of resource blocks assigned to user i. U_i is the utility concave function of user i depending on the number of resource blocks the user gets.

Assume $\{R_1, R_2, ..., R_N\}$ is the optimal solution of the above problem, we have:

$$U_{\max} = \sum_{i=1}^{N} U_i(R_i)\Big|_{R_1+R_2+...+R_N=R} \geq \sum_{i=1}^{N} U_i(R_i^{`})\Big|_{R_1^{`}+R_2^{`}+...+R_N^{`}=R} \quad \forall\{R_i^{`}\} \qquad (10)$$

Assume we have now $(R+1)$ resource blocks, we have a new problem:

$$\max\left(\sum_{i=1}^{N} U_i(r_i)\right) \quad st. \quad \sum_{i=1}^{N} r_i = R+1 \quad r_i \in Z^+ \qquad \text{(Problem II)}$$

Consider a feasible solution $\{R_1, R_2, ..., R_k+1, ..., R_N\}$ of Problem II with k is chosen such that:

$$U_k(R_k+1) - U_k(R_k) = \max_{i\in[1,N]}\{U_i(R_i+1) - U_i(R_i)\} \qquad (11)$$

The resulting total utility is:

$$U = \sum_{i=1}^{N} U_i(R_i) + U_k(R_k+1) - U_k(R_k)$$

Consider another feasible solution of Problem II: $\{R_1^{"}, R_2^{"}, ..., R_N^{"}\}\Big|_{\sum_i^N R_i^{"}=R+1}$.

a) $R_k^{"} \geq R_k + 1$

$\{R_1^{"}, R_2^{"}, ..., R_k^{"}-1, R_N^{"}\}$ is a feasible solution of Problem I, because $R_1^{"} + R_2^{"} + ... + R_k^{"} - 1 + ... + R_N^{"} = R$.

According to Eq.7,

$$\sum_{i=1}^{N} U_i(R_i) \geq U_1(R_1^{"}) + U_2(R_2^{"}) + ... + U_k(R_k^{"}-1) + ... + U_N(R_n^{"})$$

Furthermore,

$U_k(R_k+1) - U_k(R_k) \geq U_k(R_k^{"}) - U_k(R_k^{"}-1)$ because U_k is a concave function, so its marginal utility decreases with increasing R. Therefore:

$$\sum_{i=1}^{N} U_i(R_i) + U_k(R_k+1) - U_k(R_k) \geq U_1(R_1^{"}) + U_2(R_2^{"}) + ...$$
$$... + U_k(R_k^{"}-1) + ... + U_N(R_n^{"}) + U_k(R_k^{"}) - U_k(R_k^{"}-1) \qquad \forall\{R_i^{"}\}$$

or $\sum_{i=1}^{N} U_i(R_i) + U_k(R_k+1) - U_k(R_k) \geq \sum_{i=1}^{N} U_i(R_i^{"})$

$\{R_1, R_2, ..., R_k+1, ..., R_N\}$ is hence the optimal solution of the Problem II.

b) $R_k^{''} < R_k + 1$

It exists q so that $R_q^{''} > R_q$ or $R_q^{''} \geq R_q + 1$. Since U is concave and has decreasing marginal utility, we have:

$$U_q\left(R_q+1\right)-U_q\left(R_q\right) \geq U_q\left(R_q^{''}\right)-U_q\left(R_q^{''}-1\right)$$

According to Eq.8, we have:

$$U_k\left(R_k+1\right)-U_k\left(R_k\right) \geq U_q\left(R_q+1\right)-U_q\left(R_q\right) \geq U_q\left(R_q^{''}\right)-U_q\left(R_q^{''}-1\right)$$

And because $\left\{R_1^{''}, R_2^{''}, ..., R_q^{''}-1, R_N^{''}\right\}$ is a feasible solution of Problem I, following Eq.1, we have:

$$\sum_{i=1}^{N} U_i\left(R_i\right) \geq U_1\left(R_1^{''}\right)+U_2\left(R_2^{''}\right)+...+U_q\left(R_q^{''}-1\right)+...+U_N\left(R_n^{''}\right)$$

Therefore:

$$\sum_{i=1}^{N} U_i\left(R_i\right)+U_k\left(R_k+1\right)-U_k\left(R_k\right) \geq U_1\left(R_1^{''}\right)+U_2\left(R_2^{''}\right)+...+U_k\left(R_k^{''}-1\right)+...$$
$$...+U_N\left(R_n^{''}\right)+U_q\left(R_q^{''}\right)-U_q\left(R_q^{''}-1\right)$$

$\left\{R_1, R_2, ..., R_k+1, ..., R_N\right\}$ is hence the optimal solution of the Problem II.

To conclude, if there is one more available resource block, this block will be assigned to the user who has the maximum marginal utility given that the previous resource assignment is optimal.

If the total number of resource blocks is 1 (*R=1*), the optimal solution is trivial. The resource block is assigned to the user *k* such that: $U_k\left(1\right) = \max_i \left\{U_i\left(1\right)\right\}$. If *R=2*, take the solution from *R=1*, and assign the rest resource block to the user who achieves the maximum marginal utility. If *R=n*, take the optimal solution when *R=n-1*, and assign the last resource block to the user who gets the maximum marginal utility.

Pre-reservation of TV Channels to Improve the Availability of IPTV Services Offered in Vehicular Networks

Sadaf Momeni, Sonja Weichler, Bernd E. Wolfinger
Department of Computer Science,
Telecommunications and Computer Networks
University of Hamburg
Hamburg, Germany
{momeni, weichler, wolfinger}@informatik.uni-hamburg.de

***Abstract*—The increasing popularity of IP-based vehicular networks is currently leading to a strong interest in IPTV services offered to users in vehicles. For real-time communication services such as live IPTV the service quality as experienced by the users (QoE) is of particular importance. QoE of IPTV is strongly determined by the probability that a TV channel required by a user is unavailable upon its request, called channel blocking probability (CBP). Channel blocking in vehicular networks may either happen at a handover instant when a car changes its cell in the wireless access network or when an IPTV user within a cell changes the TV channel watched. Evidently, handover-induced channel blocking is particularly annoying for a user. Therefore, in this paper we introduce a novel algorithm which tries to significantly reduce the handover-induced CBP. Our algorithm relies on an a priori reservation of the currently watched channel in the neighboring cell before the user actually reaches this cell. We present comprehensive studies to investigate how strongly an a priori channel reservation is able to decrease the handover-induced CBP. The studies also quantify the price we may have to pay for a channel pre-reservation in terms of an increase in the switching-induced CBP.**

***Index Terms*—IPTV, channel availability, channel pre-reservation, vehicular networks, simulation.**

I. Introduction and Motivation

IP-based vehicular networks are becoming increasingly relevant. Among other things, this stems from the fact that, in the very near future, numerous vehicles (and not only top-of-the-range cars!) will be equipped with components allowing for a direct Internet connection via a wireless access network, e.g. based on LTE [1] or WLAN [2] technology. At the beginning, applications considered for vehicular networks were mainly traffic-safety related. More recently, researchers focused on infotainment applications, too, i.e. applications which offer information and/or entertainment services. Infotainment comprises IPTV services offered to users in vehicles. Therefore, there exists a strong demand to study the quality of IPTV services in vehicular networks. Here, service quality should be assessed mainly from point of view of the human IPTV users – an aspect which is called Quality of Experience (or: QoE for short). An important QoE measure with respect to IPTV services is the probability that TV channels are unavailable upon their demand by a user. We call the unavailability of a requested channel a channel blocking event and thus we are interested in determining the Channel Blocking Probability (CBP) as a QoE measure of utmost importance.

In the past, CBP has been evaluated for IPTV services offered via widely different access networks such as DSL [3], WiMAX [4] and Vehicular Ad-hoc Networks (VANETs) [5]. Most of those CBP evaluations were based on simulation models. Only recently have the first successful studies of CBP for IPTV services in VANETs been published which are based on analytical models [6], [7]. Blocking of TV channels in vehicular networks may have two reasons: Either, a currently watched channel may not be available because of a bandwidth bottleneck in the newly reached cell (when a car changes its cell in the cellular access network). Or, a newly requested channel may not be available (again as a consequence of bandwidth limitations) when a user within his current cell switches to a new TV channel. Evidently, handover-related blocking is particularly annoying for a TV user because the channel currently watched suddenly becomes unavailable. Therefore, an important goal when trying to improve QoE of IPTV service is to reduce the probability of handover-related blockings. To the best of our knowledge, no proposal yet exists in the literature which tries to decrease the number of handover-related blockings in vehicular networks offering an IPTV service. So, we decided to elaborate an approach to reduce the handover-related CBP. Our main idea is to reserve the channel currently watched by a user a priori in the

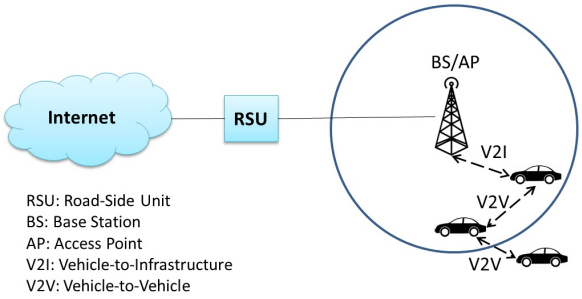


Fig. 1. General model of vehicular networks.

neighboring cell that a given car will drive into. As we consider only motorway scenarios in this paper, we know in advance the cell which will be reached next if we neglect motorway exits and intersections. In this paper, we present our novel algorithm proposing a priori reservation of TV channels and called RCBH (for: Reserve Channel Before Handover). RCBH has not yet been published before. We have embedded the RCBH algorithm into our detailed simulation model for IPTV in VANETs (cf. [5], [8]) in order to study the improvement of handover-related CBP to be expected when using RCBH. The results delivered by the extended simulation model comprise both, CBP for handover-induced channel request events as well as the probability of channel blocking for switching-induced channel requests.

The paper is structured as follows: Section II presents the required basics regarding VANETs, IPTV (architecture and user behavior modeling) as well as QoE measures. The RCBH algorithm is presented in detail in Section III. Section IV sketches the embedding of RCBH into our simulation model for CBP calculation and summarizes some of the test experiments which we have executed to verify the correctness of the extended simulation tool. Section V comprises comprehensive case studies which demonstrate how strongly the usage of RCBH can reduce the handover-related channel blocking probability and which, moreover, investigate whether and how much we have to pay for the handover-related blocking improvement in terms of an increase of the switching-related channel blocking probability. In Section VI, we summarize the results obtained and give an outlook to desirable future work.

II. IPTV Services in Vehicular Networks and Quality of Experience Measures

A. Vehicular Networks

Vehicular based multimedia applications are expected to play a very important role in the future of vehicular infotainment and intelligent transportation systems. They may contribute to traffic safety and efficiency by providing detailed information about accidents which will help in the intervention of emergency services and will allow drivers to predict the duration of traffic jams in various areas [9].

Vehicular networks support two types of communication: Vehicle to Vehicle (V2V) and Vehicle to communication Infrastructure (V2I). As shown in Fig. 1, Vehicle to Vehicle communications can be achieved by means of direct or multi-hop communication, between the vehicles. V2V communication is strongly related to Mobile Ad-Hoc Networks, as the connections are established between vehicles, which can be seen as mobile nodes. They are independent from static energy sources, because vehicles (e.g. cars) generate the needed energy by themselves. Also they are highly mobile, and only restricted by the underlying infrastructure (e.g. road). The communication between vehicles is based on DSRC and WAVE [10], [11].

In V2I, the communication is established between the vehicles and the Road-Side Units (RSUs). This communication infrastructure, via access points (APs) or Base Stations (BSs), provides services such as Internet access to the users of a vehicular network [12], [13]. The communication technique is mainly based on GPRS, WiFi, WiMAX or LTE. Altogether the combination of a fixed Roadside-Backbone-Network (RBN) composed of RSUs, and the Ad-hoc infrastructure built up by the communications between the vehicles, makes the construction of real test-scenarios for VANETs a challenge. Simulations are sufficient evaluation methods, but there also exist practical achievements for V2I communications to build-up Roadside-Backbone-Networks: At the University of Rostock an RBN architecture was elaborated including early prototype implementations assuming fixed access nodes along the roads that can be accessed by the vehicles by means of wireless communication [14]. It is also suitable to use this infrastructure for broadcasting IPTV or other multimedia services for vehicular users.

B. IPTV Services in VANETs

IPTV, as an acronym for Internet Protocol Television, offers the transmission of audio, data and video over the Internet by means of IP. As it is suggested by other authors, too [15], [16], we distinguish IPTV from Internet-TV such that users have to be logged into a separate network for receiving secured IPTV services with guaranteed QoS by the provider. Users connected to the Internet are able to receive the requested TV-Channels or Services anytime they want. To extend this to anywhere they are, we need to establish VANETs regions. Users are then able to use IPTV services even for long car journeys. So, IPTV is the next step in the development of Internet-Services

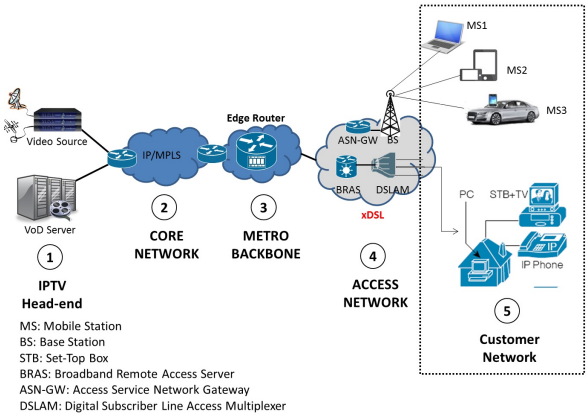


Fig. 2. IPTV system architecture.

over Vehicular Networks and an ambitious intention for researchers [17].

A typical IPTV network topology is divided into 5 main parts: IPTV head end, core network, metro backbone, access network and subscribers (cf. Fig. 2). These networks may be owned and managed by different network service providers.

At the IPTV head-end (1) live TV channels are encoded, encrypted and delivered in the form of IP multicast streams. After encoding, each channel is encapsulated into IP packets and transmitted over the core network (2) which represents a high-speed communication infrastructure. The core network distributes the video flows from the head-end to the metro backbone. The metro backbone (3) interconnects the core network with the different access networks. Basic technology of the metro backbone network is Wavelength-division multiplexing (WDM). It is needed for effective data transmission and routing of packets between the core and the access network [18].

Finally, the subscribers – which form the customer network (5) – may be connected to the access network either directly or have their own local network which enables indirect communication and information exchange between the user's device (e.g. TV set) and the access network. This direct or indirect communication allows accessing the available resources in the IPTV network.

C. Quality of Experience Measures for IPTV

There exist a lot of different QoE measures related to IPTV, including Mean Opinion Score (MOS) [19], PEVQ and PEAQ [20] to judge the video and audio quality of the TV channels as they reach the IPTV subscribers. Another important QoE measure is given by the channel switching delay [4] which should be as small as possible from the users' point of view.

As this paper focuses on the availability of IPTV services, the main QoE measure which is of interest to us is the Channel Blocking Probability (CBP) as already mentioned in Section I. As we want to distinguish between channel blocking events due to handover and those due to channel-switching within a given cell, we need two further variables HBP and SBP to characterize QoE. Let us now define those more specific QoE measures. Let $T = [t_1, t_2]$ be a time interval, where $t_2 > t_1$ and |T| = t_2 - t_1 denote the length of T.

Let further denote

- $\#e_h$(T) $\triangleq$ number of handover-events during T for a given cell.
- $\#e_s$(T) $\triangleq$ number of switching-events during T within a given cell.
- $\#b_h$(T) $\triangleq$ number of channel blockings due to handover during T for a given cell.
- $\#b_s$(T) $\triangleq$ number of channel blockings due to switching during T within a given cell.

Then, we define

$$\text{HBF(T)} \triangleq \frac{\#b_h(T)}{\#e_h(T) + \#e_s(T)}$$

i.e. handover-induced blocking frequency during T (related to all events when a channel is required for a user, either due to handover or due to switching).

$$\text{SBF(T)} \triangleq \frac{\#b_s(T)}{\#e_h(T) + \#e_s(T)}$$

i.e. switching-induced blocking frequency during T (again related to all events when a channel is required for a user).

We assume that for |T| $\rightarrow \infty$ the frequencies converge to the corresponding probabilities and thus we get:

$$\text{HBP} = \lim_{|T|\rightarrow\infty} \text{HBF(T)}$$

and

$$\text{SBP} = \lim_{|T|\rightarrow\infty} \text{SBF(T)},$$

called *Handover-induced* and *Switching-induced Blocking Probability*, respectively. And, evidently also

$$\begin{aligned}\text{CBP} = \text{HBP} + \text{SBP} &= \lim_{|T|\rightarrow\infty} (\text{HBF(T)} + \text{SBF(T)}) \\ &= \lim_{|T|\rightarrow\infty} \frac{\#b_h(T) + \#b_s(T)}{\#e_h(T) + \#e_s(T)}\end{aligned}$$

A straight-forward interpretation of this is that HBP corresponds to the portion of CBP which is due to blockings at handover events. For SBP an analogous interpretation holds and that is why HBP + SBP

TABLE I
SIMULATION VARIABLES

Notations	Descriptions
N	Number of TV Channels offered in total
C_i	TV Channel nº i
Z_i	Cell i
BW_c	Total Bandwidth available for IPTV in Cell c
FBW_c	Free Bandwidth in Cell c
NB_c	Number of different channels broadcasted or reserved in Cell c
U_n	User with ID n (e.g. IP address)
CMM_n	Channel Management for mobile users U_n
CMZ_i	Channel Management in Z_i
CBL_i	Channel Broadcast List of CMZ_i

= CBP. Besides HBP, SBP and CBP, we are also interested in the general risk for a single IPTV user to experience a channel blocking when it comes to a handover for this user; let us denote this risk by HBR, which again is determined in terms of a channel blocking probability. We define

$$\text{HBR} = \lim_{|T|\to\infty} \frac{\#b_h(T)}{\#e_h(T)}$$

and call it *handover-related blocking risk*.
Similarly, we are interested in the risk to experience a blocking event when a single user switches his/her channel. We introduce

$$\text{SBR} = \lim_{|T|\to\infty} \frac{\#b_s(T)}{\#e_s(T)}$$

which we call *switching-related blocking risk* and evidently observe that typically HBR + $SBR \neq CBP$ for $HBR < 0$ and $SBR < 0$.
In this paper, to evaluate QoE if we apply our RCBH algorithm, we urgently need HBR and SBR because in case of channel pre-reservation the risk of a channel blocking in general is different for handover and switching events.

III. A NEW ALGORITHM (RCBH) FOR PRE-RESERVATION OF TV CHANNELS TO REDUCE HANDOVER-RELATED CHANNEL BLOCKING PROBABILITIES

Due to the shortage of bandwidth in vehicular networks there is a need to develop a "Reserve Channel Before Handover" Algorithm (RCBH). Its goal is to ensure continuous availability of a channel requested for a longer period of time (called viewing-phase) by a user and therefore increase the QoE for IPTV as described in the introduction. The components in the user-devices and within the cells enable a channel-reservation between neighboring cells. Whereas zappers do not get the reservation for channels they still benefit from this algorithm once they zap to already broadcasted or reserved channels. This holds in particular if zapping to specific channels (likewise viewing them) is determined by Zipf's law which is a quite common and rather realistic assumption also made in this paper.

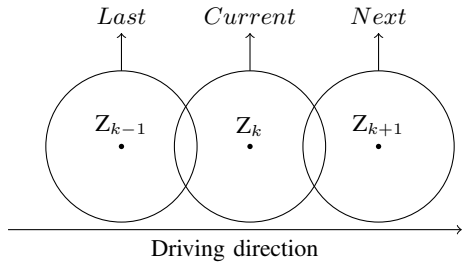


Fig. 3. Construction of cells and defining driving direction.

A. Required structures for the RCBH-Algorithm

To introduce the algorithm some preparations have to be realized. Table I is given to simplify the understanding of our text. In our description of the RCBH-Algorithm we first define what is meant by "current", "next" and "last" cell (Fig. 3) and describe roughly the components used within the algorithm before we go into detail showing the $join$, $leave$ and $reserve$ procedures to finally explain the $handover$ event. Obviously, the current cell is the one in which the user is at present. We will use Z_k referring to the current cell. Depending on the driving direction of the user the "next" cell is defined as the one for the user upcoming and adjacent to the "current" cell. Without loss of generality, we declare it as Z_{k+1}. Finally the "last" cell (in the sense of previous cell) is the one just left by the user, of course adjacent to the "current" cell as well. We will name the last cell Z_{k-1}. Please note that the next cell which a vehicle will reach can be determined in a straight-forward manner as we limit ourselves to the consideration of motorway scenarios without taking into account the (typically small number of) vehicles, leaving the motorway at exits or the alternative paths possible at motorway intersections.

For communication between IPTV users and service provider a channel-management (CM) is needed. It is a basic communication component in both, the mobile devices (CMM) and the stationary access-point in the cell (CMZ). As shown in Fig. 4, communications can be established between the CMMs and CMZ. Mainly $join(C_i)$, $leave(C_i)$ and $viewing(C_i)$ messages are sent from CMM to CMZ. Looking at it the other way around the requested channel is

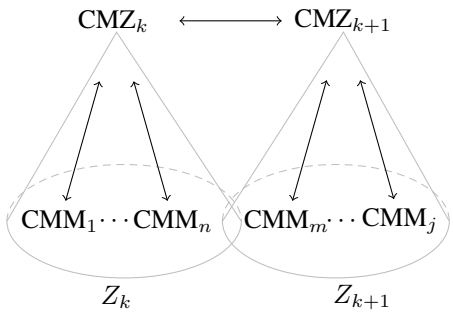


Fig. 4. Channel Management communication.

transmitted to the CMM of a user who successfully joined a multicast group for a specific channel in the CMZ. Additional communication between two adjacent radio cells' CMZ has to be implemented to mutually exchange the $reserve$-demands.

In order to define the basic events $join$, $leave$ and $reserve$, the Channel Broadcast List (CBL) is of fundamental importance.

Information about requested channels are stored in the CBL. It is used by the CMZ to save reservations and distinguish them from users who are currently in a viewing-phase and to which a channel is delivered.

The CBL consists basically of three columns as shown by way of example in Table II. All channels offered in total are listed up in the first column from 1 to N. In the next column the user-entries are saved. Here, the user's IP-address will work well as it is a unique attribute. A flag attached to the user's IP-address helps determining in which mode the user is. And finally the expiration time for reservations is stored in the third column.

This CBL is sufficient to store information about the users and to calculate the bandwidth currently still available for IPTV services. For calculation of free bandwidth in a cell c (FBW$_c$) we need the number of different channels currently broadcasted or reserved (NB$_c$). By summing up all channels with at least one entry in the second column of CBL$_c$ we obtain NB$_c$. As a result the calculation for FBW$_c$ is as follows:

$$\mathrm{FBW}_c = \mathrm{BW}_c - \mathrm{NB}_c \tag{1}$$

Obviously, NB cannot be greater than the total capacity BW$_c$ of a cell provided for IPTV service. Thus, requests for channels neither reserved nor broadcasted can only be executed if FBW is currently sufficiently large to deliver the demanded channel. When starting to broadcast a not yet requested or reserved channel in Z$_c$ the FBW$_c$ will decrease as a result of an increasing NB$_c$. The recalculation of FBW can be done directly after finishing the handling of an event. This kind of modification of the RCBH-Algorithm is shown in Fig. 5 line 11, where the join process is explained.

TABLE II
CHANNEL BROADCAST LIST

Channel$_{ID}$	Recipient$_{ID}$ [flag]	Expiration of reservation
001	IP_address$_7$ [R]	12:23
	IP_address$_2$ [V]	-
002	-	-
003	IP_address$_{10}$ [Z]	-
	IP_address$_4$ [R]	12:25
004	IP_address$_5$ [R]	12:30
...	...	...
100	IP_address$_{23}$ [Z]	-
...	...	...

Meaning of the Flags:
R: user has reserved this channel
V: user is viewing the requested channel
Z: user is in zapping mode, watching this channel for a short time

B. Description of the CBL flags

The flags in the CBL indicate the states users take up in the current cell. Three states are possible and will be saved in the user-entry column. They are denoted as zapping [Z], viewing [V] and reserved [R]. Zapping and Viewing describe a state for a user in the current cell, whereas [R] is a reservation-state-flag in the next cell's CMZ. A reservation will remain valid according to the period of time (expiration-time) stored in the third column of the CBL. It is important to point out that the possible state-sequences are inter-dependent. In fact only the following state-sequences are possible when first-time joining a channel is completed: [Z]$\vee$([Z],[V]) $\vee$ ([Z],[V],[Z]) and so on. As long as the user sufficiently quickly switches to other channels, the user will be considered to be zapping. If the user is in the zapping mode, as already explained, no reservation is demanded in the next cell. As a consequence, zappers do not benefit from reserving channels in the next cell since they switch too quickly between the channels. Zappers in small cells with high request-frequency and -density for different channels get even more vulnerable with respect to channel-blocking.

C. Elementary events influencing the CBL

As shown in Table II the flags can be one of [Z], [V] or [R]. These are caused by certain events. The elementary events in the RCBH-Algorithm are referred to as $join$ and $leave$.

1) Joining a channel: Joining a channel means, that the CMM sends a $join$ request to the current CMZ and after internal calculation the channel is

transmitted to the requesting user if currently still a sufficient amount of free bandwidth exists or the requested channel is currently already multicasted. This $join$-request does include the desired Channel nº (C_i) and of course a unique user ID (U_n). What happens after receiving a $join$-request inside a particular CMZ_k is presented in Fig. 5. In line 1, the current CMZ_k receives a $join$(C_i, U_n)-request from the user's CMM_n. The CMZ_k first checks, whether the user already successfully reserves this channel (line 2). If this turns out to be true, the existing reserving-entry in the CBL_k will be corrected by changing the [R] into a [V] flag and deleting the reservation-expiration time. Additionally, a $reservation$-request will be sent to CMZ_{k+1} (next cell's CMZ) by the current cell's CMZ_k. If no reservation for that user in the CBL_k was realized, it is possible, that the channel is already broadcasted or reserved for other users (U_*), so that further entries in the CBL_k for broadcasting this channel do not increase the NB_k (line 6). In this case, the user will be added with a zapping flag (line 7). If furthermore no entry (neither broadcast nor reservation) was made for the requested channel in the CBL_k, the CMZ_k yet has to check whether a sufficient amount of FBW_k in Z_k is left to transmit C_i. In case enough FBW is left, the CMZ_k adds the user to CBL_k with a zapping-flag [Z] and moreover it increases NB_k and recalculates FBW_k (line 8-11). In all other cases, C_i will not be broadcasted and thus be blocked. Joining a C_i in Z_k, which is not already broadcasted or reserved, is accompanied by a reduction of free bandwidth (FBW_k) in the current cell. Obviously the FBW-recalculation is of high importance. More FBW will be generated by users, who $leave$ the cell or switch (as the last user of one channel) to another channel.

2) Leaving a channel: Besides $join$, $leave$ is an elementary event. It is used whenever the user $leaves$ a channel. This will happen, e.g., every time the user zaps to another channel, he/she leaves a cell (i.e. handover process) or switches off the device. When leaving a channel, the CMM_n sends a $leave$-message to the current CMZ_k. In that case, the CMZ_k looks up the CBL_k and removes respective entries. In the case, that the user was in zapping-mode, the entry will be deleted. $Leaving$ a channel as the last user increases the cell's FBW. If the user however was in viewing-mode and the [V] flag was set, then the current CMZ_k also sends a cancel-reservation-signal (e.g. $leave$) to the next CMZ_{k+1}. And thus, if the only entry for that specific channel is removed in both cell's CBLs, then FBW_k and FBW_{k+1} are increased by the capacity of one channel. Again, the recalculation of FBW is very important for the RCBH-Algorithm.

Require: join request from CMM
1: Receive a $join$(C_i, U_n)-request from CMM
2: **if** Corresponding reservation-entry in CBL existing **then**
3: Change flag from [R] to [V]
4: Delete the reservation-expiration time in CBL
5: Send a $reservation$(C_i,U_n)-request to the next CMZ
6: **else if** Other users receive/reserve C_i **then**
7: Add U_n[Z] to the desired channel C_i
8: **else if** there is enough FBW left in the Cell **then**
9: Add U_n[Z] to the desired channel C_i
10: Raise NB by 1: NB++
11: Recalculate FBW
12: **else**
13: Block desired channel
14: **end if**

Fig. 5. CMZ: process $join$-request

D. Changing flags in neighboring CBLs

Of rudimentary importance for pre-reservation is a predefined time-slot ΔT which is referred to as a minimum of one minute in [12]. It defines a period of time after which the user reaches the viewing-phase. However, if no zapping has been made within ΔT the user's CMM sends a viewing-flag-message to the current CMZ to indicate, that the user is now in a viewing-phase. Along with the message sent by the CMM, the CMZ changes the state-flag behind the user ID from [Z] to [V]. Reaching the viewing-phase by exceeding the threshold ΔT is calculated by CMM to avoid getting no reservation at all.

As described above, there is no way for a user to get from the [Z] state to the [R] state immediately. For example the user is zapping and after a while the user decides to remain at a channel. As described, after ΔT the user enters the viewing-phase and the current CMZ performs a channel-reservation in the next cell. Nevertheless, the user can choose to change the channel again, so that the flag in the current CBL changes from [V] to [Z]. Besides changing the flag in the current CBL the algorithm also has to cancel the reservation in the next CBL. This has a positive impact on the FBW-calculation in the next cell. If the user is not changing the channel in the current cell, the reservation must remain until the user enters the next cell. Then, however, the reservation-flag is converted to a viewing-flag and like a chain-reaction the current cell sends a reservation-request to the next cell and so on.

Receiving a leave-request from the CMM, the current cell's CMZ transmits a reservation-request to the next cell's CMZ. This message contains the user's IP-address and the desired channel number.

After receiving the reservation message the next cell's CMZ adds an expiration time (E) to the CBL user-entry after which the reservation will be canceled. The reservation is done only once namely if and only if the user accomplishes the viewing-state. The reservation [R] is exclusively issued by the current cell's CMZ whereas the reservation-request is generated by the last cell's CMZ. Reserving is similar to broadcasting a channel. If the channel is not yet broadcasted or reserved first reservation- or join-requests lead to an increase of NB and a reduction of FBW. To prevent a channel to be reserved too long and thus eventually blocking free bandwidth capacity in the next cell, the algorithm decides whether a reservation is not needed anymore. One scenario in which a reservation should evidently be canceled, is when the user takes a break in a parking bay but still needs the availability of the requested channel in the current cell. Since the reservation already proceeded, as described above, it is unprofitable to reserve the channel in the next cell for as long as the break may take. The second scenario occurs when the user is part of a traffic jam. Like the parking bay example, we do not know, if the user makes use of the reservation in the next cell nor when he will. Thus blocking free bandwidth for an unknown period of time should be avoided. To believe in future traffic science there will be software implemented in cars, that will prevent or at least reduce traffic jams. In avoidance of reserving for an endless time-interval we can calculate the latest possible time at which the user is expected to enter the next cell. We assume that there is a recommended minimum speed attached to the lanes. Since highways may only be used with vehicles whose specific design speed is more than 60 km/h we divide the cell-size by the speed of 60 km/h. If desirable we could enlarge the duration of reservations by means of halving the speed or doubling the cell-size, the result is the same.

E. Combined Events - causing changes in cross-cells CBLs

Up to now we demonstrated, how the elementary events modify the CBLs of two adjacent cells. The join process only manipulates the current cell's CBL, whereas the leave event may change the neighboring cell's CBL as well (in case of canceling a reservation) during handover. Also changes caused by the viewing behavior of the user are possible.

Requiring a reservation-entry in the next cell's CBL, the $handover$-process starts right before leaving the current cell. It is a complex event combining $join$, $reserve$ and $leave$ in two adjacent cells. The first requirement is a successful access to a channel, where the flag has to be V in the current CBL. Upon successful joining of a channel, the current CBL sends the reservation-request to the next CBL. For a smooth handover, the reservation has to be fulfilled in the next cell. It should be noted that the technical details of how the handver is actually performed as well as its duration are not of much interest to us, because we expect the impact this has (on CBP and HBP) should be negligibly small. From the user's point of view, the handover process is described as follows. In the first step, the CMM recognizes the presence of the reserving cell. Precisely at the moment, when it comes to join the next cell, the CMM sends a $join$ to the current (the reserving cell) and a $leave$-message to the last cell, the user just attended. After recognizing the reservation in the current CBL, the affected CMZ is changing the reservation-flag to a viewing-flag. In that case, as already described, the CMZ sends a reservation-request to the next cell. In total, we formally have to change the formerly $current$ to $last$ and the $next$ to $current$ cell. In other words, a handover can be split up in a $join$- and a $leave$-request. Also this is important for the FBW recalculation of both, the current and the next cell, which are referred to last and current cell after the effective handover-event. A successful handover is thus a sequence of the following steps:

1) join the current cell Z_k
2) leave the last cell Z_{k-1}
3) reserve in the next cell Z_{k+1}

In each of the steps, the FBW-recalculation is of high importance. In the following section we have a detailed look into the embedding of the algorithm into a simulation tool followed by some test experiments showing the correctness of our RCBH simulator.

IV. Embedding of RCBH into a Simulation Model and Test Experiments for this Simulation Tool

To evaluate the RCBH algorithm used for decreasing HBP and in general CBP, we have developed a Monte Carlo simulation model covering the algorithm proposed in Section III. We elaborated a dedicated simulator to simulate highway scenarios.

Comprehensive simulation experiments have been carried out with the parameter values presented in Table I, IV, V, and the typical configuration underlying the simulation model is demonstrated in Fig. 6. In our simulation model, vehicles entering the cell at the border of the geographical area (GA) are assumed to contain either 0 or at least 1 passenger watching TV (of course, the driver should not watch!). TV is not switched on or off in a car during the car journey along the complete highway sector observed.

We elaborated a simulation tool for analyzing IPTV services offered in communication networks of VANET type. This simulation tool has been designed to generate a system model abstracting from the

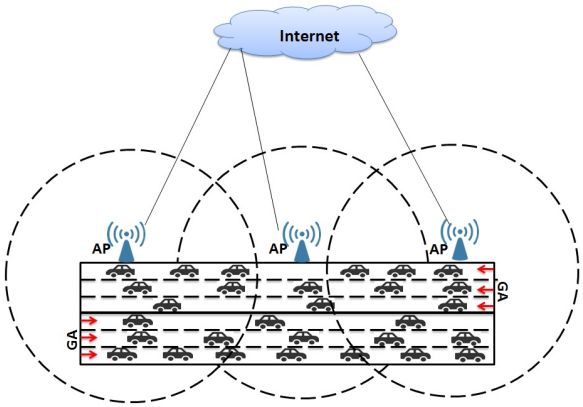


Fig. 6. Motorway scenario based on 3 neighboring cells in which IPTV service is offered.

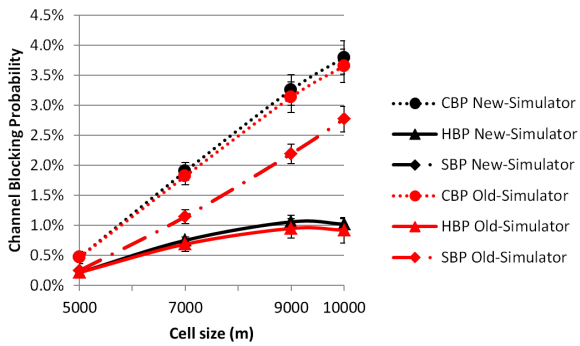


Fig. 7. Channel blocking probability against different cell sizes.

lower layers of the protocol stack because they are not relevant to the QoE measures studied. However, unlike other VANET simulators, our tool offers a very detailed model to specify IPTV user behavior, because modeling user behavior in a realistic manner is highly relevant for our type of studies. Our simulator is written purely in C and we are using the LoadSpec tool to generate aggregate traces of channel switching events for the vehicular IPTV users. LoadSpec has been elaborated in our TKRN research group [21]. It is an artificial load generator for different interfaces, and it is capable of producing realistic network traffic with different characteristics in a very simple and flexible manner. Other simulators for VANETs such as ns-2 [22], ns-3 [23], JiST/SWANS [24], GloMoSim [25] and OMNET++ [26] emphasize on the lower protocol layers as they have completely other research goals. Doing availability studies based on those simulators would be extremely inefficient, if possible at all. Therefore, we are using our own tool for evaluating the availability of IPTV services.

We model the probability of a TV channel being selected by a user with the Zipf distribution [27] which has an ability to represent the skewed popularity distribution of objects. In the experiments, the request probability P_i of the i-th popular channel is determined by the Zipf distribution and calculated by eq. (2).

$$P_i = \frac{1/i^{\Theta}}{\sum_{k=1}^{N} (1/k^{\Theta})}, \qquad (2)$$

where N is the total number of distinct channels, k is their rank and θ is the Zipf parameter that determines the degree of popularity skew. When $\theta = 0$, all channels are equally popular. As the value of θ increases, the popularity of channels is increasingly skewed. For θ, we choose a value of 1.3 which is realistic according to measurements of IPTV user behavior, cf. [4].

After we had implemented our extended IPTV simulator taking into account our new RCBH algorithm, we decided to carry out a very thorough testing phase for this simulator. As no measurements regarding real vehicular networks which offer IPTV services have been available for us up to now model validation for the new simulator is impossible and this makes thorough testing indispensable from our point of view.

The following approaches for testing of our simulator were therefore applied by us in combination:

1) an in-depth code inspection covering the whole simulation program which led to a very thorough static debugging of the simulator;
2) checking of important partial results determined during the simulation experiments, such as the values of $\#e_h(T)$ and $\#e_s(T)$, which can also be determined directly by means of simple analytical calculations;
3) carrying out a lot of plausibility tests regarding the simulation results obtained;
4) and, last but not least, we carried out test case studies in which we used the originally existing IPTV simulator (as presented, e.g., in [5], [8]) and we compared the results obtained by this simulator with the results delivered by our new, extended simulator covering RCBH. In order to fairly compare both simulators, for the RCBH simulator we used a version in which the a priori reservation of TV channels in neighboring cells is only executed with a probability β. And for the comparison studies we used the results of the RCBH simulator for $\beta \to 0$.

Let us now shortly summarize some of the results of the test case studies as mentioned in point 4) of the list. In the scenarios described we made the same assumptions in both simulation tools and throughout the comparison experiments we varied the cell size.

TABLE III
PARAMETER VALUES FOR THE TEST COMPARISON CASE STUDY.

Notations	Descriptions	Values
N	Number of TV channels offered in total	100
BW_c	Total Bandwidth available for IPTV in Cell c	20 Mb/s (sufficient for parallel transmission of 40 TV Channels)
D_{cell}	Diameter of each Cell	5000, 7000, 9000, 10000 m
k	Number of lanes in each direction of the highway	3
$SP_{[i]}$	Average speed of a vehicle in each lane i per direction [km/h]	90 (lane 1), 120 (lane 2), 150 (lane 3)
$d_{min[i]}$	Minimum distance in lane i between two adjacent vehicles [m]	10 (lane 1), 15 (lane 2), 20 (lane 3)
$d_{avg[i]}$	Average distance in lane i between two adjacent vehicles [m]	20 (lane 1), 25 (lane 2), 30 (lane 3)
$d_{max[i]}$	Maximum distance in lane i between two adjacent vehicles [m]	30 (lane 1), 35 (lane 2), 40 (lane 3)

All our assumptions for this comparison are given in Table III. In the test experiments, we aimed to find out whether the simulation tool to model our new proposed RCBH algorithm is close enough to the original simulation tool if we vary the cell size, which corresponds to varying the characteristics of the wireless access network via which the IPTV service is offered. Fig. 7 shows the CBP, HBP and SBP values for our two simulation tools as a function of the different cell sizes (in $[\mathrm{m}]$). Again, the results are given with 95% confidence intervals. The simulation results demonstrate that we obtain nearly perfect agreement between the different simulation tools and therefore the results gained by our new simulation tool are quite acceptable. Additional test comparison studies showed similar results.

V. CASE STUDIES TO ASSESS THE EFFICIENCY OF THE RCBH ALGORITHM

In order to evaluate the performance of our proposed RCBH algorithm for providing QoE for handover users during the car journey, we will look at Blocking Probability and Blocking Risk measures characterizing the QoE. The experimental results are based on two different scenarios. Two case studies are conducted, aiming to investigate the Call Blocking Probability (CBP), Switching-related Blocking Risk (SBR) and Handover-related Blocking Risk (HBR) with applying the RCBH algorithm.

A. Case Study I

In the first case study, we want to provide decision support for using RCBH algorithm for reserving bandwidth for users being in a viewing phase. In this case study, we consider medium and low traffic density with various numbers of vehicles per km. It should be noted that the terms low and medium traffic density are of course relative and we use them in comparison with a traffic jam scenario, which then would represent a high traffic density (if we only distinguish three classes of traffic densities). All our assumptions for case study I are given in

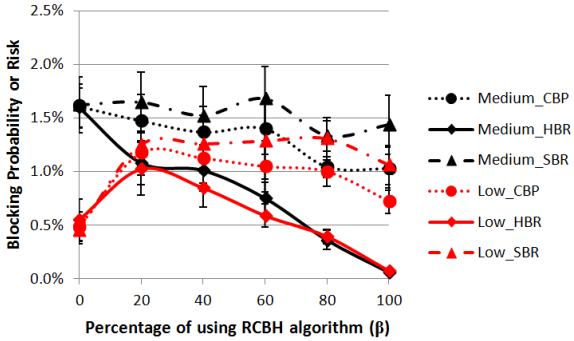


Fig. 8. CBP, SBR and HBR results against different percentage of using the RCBH algorithm.

Table IV. The distance d between vehicles is drawn according to a truncated exponential distribution for an interval with speed-dependent bounds (for details, cf [5]). We want to evaluate the impact of performing different percentage of β, i.e. the probability of applying RCBH, and vehicle traffic densities in highway scenarios on blocking probability or risk. The probability β is realized in the following straight-forward manner, i.e. whenever RCBH could be applied for an arbitrary IPTV user it is only applied with probability β. Moreover, there are two important questions that we want to answer:

- Q1. What happens when we reserve the channels only for some slight percentage of the IPTV users?
- Q2. What value of β is providing the best QoE for watching the TV programs smoothly and without loosing a currently watched program during the handover process?

We execute a set of experiments in order to answer the above questions. Fig. 8 demonstrates the simulation results (with 95% confidence intervals), which are plotted as the evaluated CBP, SBR and HBR against different values of β (ranging from 0% to 100%, with the step-size of 20%).

The observations from Fig. 8 are listed below:

- When the traffic density increases, CBP, SBR and HBR also grow and the more users are in a

TABLE IV
PARAMETER VALUES FOR CASE STUDY I.

Notations	Descriptions	Values
N	Number of TV channels offered in total	100
BW_c	Total Bandwidth available for IPTV in Cell c	20 Mb/s (sufficient for parallel transmission of 40 TV Channels)
D_{cell}	Diameter of each Cell	10 km
k	Number of lanes in each direction of the highway	3
M-$\mathrm{SP}_{[i]}$	Speed of a vehicle in each lane i [km/h] for Medium traffic density on highway	80 (lane 1), 100 (lane 2), 120 (lane 3)
L-$\mathrm{SP}_{[i]}$	Speed of a vehicle in each lane i [km/h] for Low traffic density on highway	90 (lane 1), 120 (lane 2), 150 (lane 3)
M-$\mathrm{d}_{[i]}$	Average distance in lane i in Medium traffic density between two adjacent vehicles [m]	30 (lane 1), 37 (lane 2), 45 (lane 3)
L-$\mathrm{d}_{[i]}$	Average distance in lane i in Low traffic density between two adjacent vehicles [m]	35 (lane 1), 45 (lane 2), 50 (lane 3)
α	percentage of the IPTV users	10%
θ	Zipf parameter	1.3
β	percentage of applying the RCBH algorithm	(0, 20, 40, 60, 80, 100)%

cell, the more blocking events occur.

- When we use the RCBH algorithm to reserve the currently watched channel a priori for all IPTV users (β=100%), the risk of a channel being blocked during the handover process is near zero. This of course implies that the RCBH does increase the QoE for both traffic scenarios, i.e. low and medium traffic density. Thus, question Q1 can be answered.
- SBR and in general CBP is decreasing by using the RCBH algorithm in medium traffic density situations, and the results demonstrate that applying the RCBH algorithm for all users will imply that the blocking probability and blocking risks will nearly be eliminated in a medium traffic density scenario. Because the popular channels are already reserved by users in the viewing phase, the zapping users will not be blocked selecting channels among the reserved channels. Since we use the Zipf parameter for all simulated users (both, during viewing phase and zapping phase) there is a high likelihood for zapping users, to choose already broadcasted and thus reserved channels. The result is a reduction of SBR and thus SBP.
  But in low traffic density, zapping users are not experiencing the same situation as in a medium traffic density situation. Channels reserved or viewed, are not necessarily the most popular channels and there is a chance for broadcasting unpopular channels in the case of fewer users in the cell.

In terms of answering the question Q2, comparing both methods (reserving bandwidth or without reserving bandwidth), we do have to differ between users in zapping and in viewing phases. We want to keep the users viewing a channel for a longer period of time, thus the value of HBR is relevant to us. Reducing this value is of high importance for increasing the QoE for users in a viewing phase.

- From the results we can conclude, that for a medium traffic situation, β should be at least 20%. Increasing the percentage of reserving channels goes hand in hand with a reduction of HBR, which is caused by the fact, that there is a high percentage, that only unpopular channels are blocked. For reservations beyond 75% the RCBH algorithm also reduces SBR and CBP in medium traffic situations.
  For low density traffic scenarios, in Fig. 8, it is also obvious that the percentage of using the RCBH algorithm has to be at least 75% to reduce the HBR. If we reserve the channels for all users, more of the popular channels are reserved and thus will not be blocked. However, if we only reserve a small amount of bandwidth (e.g. $\beta = 20\%$) for oncoming users (which is to be equated with reserving bandwidth for channels received by users in a viewing phase), there is a non-negligible probability of choosing unpopular channels by zapping users. And to make it even worse, if a user in viewing phase chooses an unpopular channel, this channel will block FBW in the next cell. As a conclusion, if we increase β we get the best QoE.

B. Case Study II

In this section, we will run another case study (II) with performing the RCBH algorithm for all IPTV users and thus a fixed β=100%. The goal of case study II is to analyze the impact of different cell sizes and vehicle speeds on the CBP, SBR and HBR results.

Except cell diameter D_{cell}, the speed $\mathrm{SP}_{[i]}$ and the distance between the cars $\mathrm{d}_{[i]}$, all other simulation

TABLE V
PARAMETER VALUES FOR CASE STUDY II.

Notations	Descriptions	Values
D_{cell}	Diameter of each Cell	5, 15 km
$SP_{[i]}$	Speed of a vehicle in each lane i [km/h] V_{avg} (V_1, V_2, V_3)	**60** (50, 60, 80), **80** (70, 80, 100), **100** (80, 100, 140), **120** (100, 120, 160), **140** (120, 140, 180)
$d_{[i]}$	Average distance in lane i between two adjacent vehicles [m] (d_1, d_2, d_3)	(20, 40, 40)
β	percentage of applying the RCBH algorithm	100%

parameter values are the same as in the first case study. The new parameters are given in Table V. In this case study, we varied speed and distance between the cars such that in each experiment the number of the cars in a cell in general is constant even for two different cell sizes.

In the second case study, we try to answer another two questions for the IPTV service provider:

- Q3. What is the impact of the cell size and consequently the number of users in each cell on CBP, SBR and HBR results?
- Q4. What is the impact of the vehicle speed on the blocking probability and different blocking risks with a given fixed number of IPTV users in each cell? Evidently, vehicle speed is strongly determined by the current traffic situation and not so much by the type of vehicle observed nor by the personality of the driver.

In Fig. 9, CBP, SBR and HBR are plotted against the growth of average speed [km/h]. Again, all the simulation results included for case study II are with 95% confidence intervals.

- We observe that, with increasing the cell size the number of IPTV users will also increase. The values for small cells (e.g. 5km) are below the calculations for cells with a size of 10km. The users spend more time in each cell and therefore have more time for zapping through the IPTV channels in each cell. To answer Q3, it is obvious that in bigger cells a higher user appearance is caused, which increases the switching-induced blocking risk and the blocking probability.
- Increasing the lane-speed reduces the CBP, HBR, and SBR values considerably. We can see that if we rise the average speed each user has less time to spend in each cell. So accordingly they have less switching requests in each cell. Therefore, the CBP and SBR will be decreased by increasing the average speed. The answer for Q4 is, that with increasing speed, the overall blocking is reduced.

Very gratifying is the fact, that using the RCBH algorithm reduces the probability for handover blockings to nearly zero.

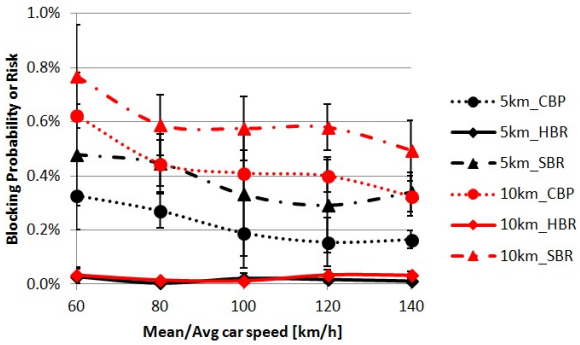


Fig. 9. CBP, SBR and HBR results against different average speed.

VI. SUMMARY AND OUTLOOK

The paper has tackled the difficult and challenging problem of reducing the probability for handover-induced blockings in IPTV services offered to vehicular users. Our approach has been based on the idea to execute an a priori reservation of currently viewed TV channels in the neighboring wireless network cell for vehicles driving on a motorway. Our algorithm RCBH to reserve channels already some time before the actual handover has been presented in detail in this paper. RCBH has been embedded in an existing simulator to determine both, handover-related as well as switching-related channel blocking risk. Comprehensive case studies have shown that the RCBH algorithm is indeed able to significantly reduce the probability of handover-induced channel blockings and thus it strongly improves QoE as observed by an IPTV user. The case studies cover different traffic scenarios, access networks based on different wireless technologies as well as different offers of TV channels. Unexpectedly, in quite a few scenarios, RCBH not only reduces the handover-related but also the switching-related blocking risk. This is a highly pleasing result because it shows that RCBH quite often implies a win-win situation as it can reduce the number of handover-related blockings without having to pay a price for this in terms of an increased number of switching-related blockings.

Of course, we should mention that the fact of restricting oneself to motorway scenarios (though it is an important use-case for IPTV) leads to assumptions

which are quite favorable for the RCBH algorithm (e.g. regarding the rather simplified mobility model as compared to general traffic situations).

Our future work will try to determine those scenarios in which the positive effect on the handover-induced blockings when applying RCBH comes along with some increase of the switching-related blocking risk. Moreover, we plan to adapt the current IPTV user model as it is applied in our case studies (which is based on measurements of user behavior of non-mobile IPTV users) to new measurements to characterize the behavior of mobile IPTV users (e.g. in VANETs) which hopefully will become available in the near future. We will also try to integrate the RCBH algorithm in real IPTV systems for vehicular networks as soon as those systems will be available which will enable us to validate our currently used simulation models.

REFERENCES

[1] R. Bachl, P. Gunreben, S. Das, and S. Tatesh, "The long term evolution towards a new 3gpp* air interface standard." *Bell Labs Technical Journal*, vol. 11, no. 4, pp. 25–51, 2007.

[2] A. K. Salkintzis, C. Fors, and R. Pazhyannur, "WLAN-GPRS Integration for Next-Generation Mobile Data Networks." *IEEE Wireless Commun.*, vol. 9, no. 5, pp. 112–124, 2002.

[3] J. Lai, *Evaluation and Improvement of TV Channel Availability for IPTV Services*. Shaker Verlag, Aachen, Oct. 2012.

[4] A. Abdollahpouri, *QoS-aware Live IPTV Streaming Over Wireless Multi-hop Networks*. Shaker Verlag, Aachen, Aug. 2012.

[5] S. Momeni, J. Lai, and B. E. Wolfinger, "Availability Evaluation of IPTV Services in Roadside Backbone Networks with Vehicle-to-Infrastructure Communication," *IEEE Conference IWCMC 2013*, 2013.

[6] B. E. Wolfinger, E. Baez, and N. R. Wilzek, "A Generalized Approach to Predict the Availability of IPTV Services in Vehicular Networks Using an Analytical Model," *14th International Conference on Networks (ICN)*, pp. 19–24, Apr 2015.

[7] B. E. Wolfinger, A. Hübner, and S. Momeni, "A Validated Analytical Model for Availability Prediction of IPTV Services in VANETs," *MDPI Electronics-Open Access Journal*, vol. 3, no. 4, pp. 689–711, Dec 2014.

[8] S. Momeni and B. E. Wolfinger, "Availability evaluations for IPTV in VANETs with different types of access networks," *EURASIP Journal on Wireless Communications and Networking, Springer Open Journal*, vol. 2014:117, 2014.

[9] J.-S. Park, U. Lee, S. Y. Oh, M. Gerla, and D. S. Lun, "Emergency related video streaming in VANET using network coding," *3rd International Workshop on Vehicular Ad-hoc Networks*, pp. 102–103, 2006.

[10] N. Loulloudes, G. Pallis, and M. D. Dikaiakos, "The Dynamics of Vehicular Networks in Urban Environments," *CoRR*, vol. abs/1007.4106, 2010.

[11] C. Huang, Y. P. Fallah, R. Sengupta, and H. Krishnan, "Adaptive intervehicle communication control for cooperative safety systems," *IEEE Network*, vol. 24, no. 1, pp. 6–13, 2010.

[12] S. Momeni and B. E. Wolfinger, "Availability of IPTV Services in VANETs Using Different Access Network Technologies," *13th International Conference on ITS Telecommunications, ITST 2013*, 2013.

[13] M. Krohn, R. Daher, M. Arndt, and D. Tavangarian, "Aspects of roadside backbone networks," *1st International Conference on Wireless Communication, Vehicular Technology, Information Theory and Aerospace and Electronic Systems Technology (Wireless VITAE)*, pp. 788–792, May 2009.

[14] A. Gladisch, R. Daher, M. Krohn, and D. Tavangarian, "OPAL-VCN: Open-Air-Lab for vehicular Communication Networks," *WiMob'10*, pp. 555–561, 2010.

[15] L. Blasco-Arcas, B. Hernandez-Ortega, and J. Jimenez-Martinez, "How to achieve customer participation and involvement in IP television," *12th International Research Conference in Service Management*, 2012.

[16] Y. Zhang, H. Z. W. Chen, and X. Gu, "A Time-Topic Coupled LDA Model for IPTV User Behaviors," *IEEE Transactions on Broadcasting*, vol. 61, no. 1, pp. 56–65, 2015.

[17] S. Park, S. H. Jeong, and C. Hwang, "Mobile IPTV Expanding the Value of IPTV," *Seventh International Conference on Networking (ICN)*, pp. 296–301, April 2008.

[18] J. Lloret, A. Canovas, J. Rodrigues, and K. Lin, "A Network Algorithm for 3D/2D IPTV Distribution using WiMAX and WLAN Technologies," *Multimedia Tools and Applications*, vol. 67, no. 1, pp. 7–30, November 2013.

[19] J. U. Duque and N. Gomez, "Quality assessment for video streaming P2P application over wireless mesh network," *The XVII Symposium of Image, Signal Processing, and Artificial Vision (STSIVA)*, pp. 99–103, September 2012.

[20] T. Chen and R. Rao, "Audio-visual integration in multimodal communication," *Proceeding of the IEEE*, vol. 86, no. 5, pp. 142–149, May 1998.

[21] A. Kolesnikov, "UniLoG: A Unified Load Generation Tool," *16th International GI/ITG Conference, MMB and DFT*, 2012.

[22] "The network simulator ns2," http://www.isi.edu/nsnam/ns/.

[23] "The network simulator ns3," https://www.nsnam.org/.

[24] "Jist/swans: Java in simulation time/scalable wireless ad hoc network simulator," http://jist.ece.cornell.edu, 2004.

[25] J. Martin, "Global mobile information systems simulation library," *UCLA Parallel Computing*, 2001.

[26] A. Varga, "Omnet++ discrete event simulation system," http://www.omnetpp.org/doc/manual/usman.html.

[27] M. Newman, "Power laws, Pareto distributions and Zipf's law," *Contemporary Physics*, vol. 46, no. 5, pp. 323–351, 2005.

Auswertung von QoS-Messdaten in einer Cloud-Umgebung

Helge Jürgensen
Nextragen GmbH
Flensburg, Deutschland
e-Mail: helge.juergensen@nextragen.de

Tadeus Uhl
Fachhochschule Flensburg
Institut Kommunikationstechnologie
Flensburg, Deutschland
e-Mail: tadeus.uhl@fh-flensburg.de

Abstract*—In der letzten Zeit gewinnt der Begriff QoS/QoE immer mehr an Bedeutung. Zu diesem Thema hat das EU-Parlament bereits im Jahr 2009 zwei Sammlungen an Direktiven veröffentlicht. Darauffolgende Forschungsprojekte der EU unterstreichen die Empfehlungen nach einer regelmäßigen Messung der QoS bei digitalen Diensten. Wie bei den digitalen Diensten selbst gibt es auch für die QoS-Bestimmung Ansätze, die Messungen in die Cloud zu verlagern. Im Rahmen dieser Arbeit sollen entsprechende Ansätze und Implementierungen von QoS-Messsystemen für Triple Play Services vorgestellt werden. Bereits vorhandene Umsetzungen bieten auch bei geringem Aufwand eine detaillierte Auswertung zur Einschätzung der Dienstgüte. Der aktuelle Stand diesbezüglich wird an konkreten Beispielen erläutert. Zum Schluss der Arbeit wird auf die mögliche weitere Entwicklung eingegangen.

I. EINFÜHRUNG

Der Begriff Quality of Service (QoS) spielt eine entscheidende Rolle in den heutigen digitalen Netzen. Der Begriff ist allgegenwärtig und wurde bereits bei der Definition der Next Generation Networks im ITU-T Standard Y.2001 [1] festgeschrieben. Am 25. November 2009 haben das Europäische Parlament und der Europäische Rat das sogenannte Kommunikationspaket angenommen, das mit den Direktiven 2009/140/EC [2] und 2009/136/EC [3] ebenfalls auf die Wichtigkeit von QoS hinweist. QoS und QoE (Quality of Experience) sollten in modernen Netzen kontinuierlich und am besten automatisiert überwacht werden. Mit diesem Ziel beschäftigen sich außerdem bereits EU-Forschungsprojekte (z.B. Leone [4] und mPlane [5]) und Standardisierungsorganisationen wie die Internet Engineering Task Force (IETF). Durch die dynamische Struktur des Internets ergeben sich besondere Herausforderungen, wenn eine flächendeckende Überwachung der Dienstgüten realisiert werden soll. Es kommen immer wieder neue Dienste hinzu, alte Dienste werden eingestellt und Rechenzentren werden umstrukturiert. Die IETF versucht diese Herausforderungen mit einem allgemeinen Framework zum Large-Scale Measurement of Broadband Performance (LMAP [6]) zu meistern und bietet damit ähnlich wie das EU-Forschungsprojekt mPlane einen allgemeingehaltenen Lösungsvorschlag an.

Diese Frameworks beinhalten zwar Grundkonzepte und Spezifikationen für die verschiedenen Bereiche eines verteilten Messsystems, wie z.B. den Austausch und die Speicherung der erhobenen Messdaten, aber sie stellen kein fertiges System dar. Es müssen erst jeweils passende Messsonden, Datenspeicher und Ergebnis-Auswertungen von entsprechenden Anbietern zusammengefügt werden. Diese auf Big-Data ausgelegten Systeme passen zum allgemeinen Trend, immer mehr Dienste in die Cloud [7] zu verlagern, da die einzelnen Komponenten ebenfalls flexibel in der Cloud arbeiten können. Neben solch großen, komplexen und sehr allgemein gehaltenen Systemen sieht man aber auch einen Bedarf an konkreten, spezialisierten Lösungen zur Überwachung von Dienstgüten, die leicht zu handhaben sind. So können auch bestehende Messsysteme für Triple Play Services von den neuen Möglichkeiten des Cloud-Computing profitieren. Da für Cloud-Lösungen im Allgemeinen keine aufwendigen Einrichtungsarbeiten beim Kunden durchgeführt werden müssen und die Kosten durch On-Demand-Lösungen geringer sind als langfristige Investitionen, können hierbei die Einstiegshürden gesenkt und der Einsatz von benötigter Messtechnik attraktiver gemacht werden. Denn trotz der konkreten Empfehlungen zur regelmäßigen Messung der Dienste-Qualität wird dies noch nicht flächendeckend umgesetzt.

Im Rahmen dieser Arbeit wird neben entsprechenden Konzepten auch eine bestehende konkrete Implementierung vorgestellt, die auf die Bestimmung von Dienstgüten bei Triple Play Services spezialisiert ist.

II. QOS-MESSSYSTEME IN EINER CLOUD-UMGEBUNG

Es gibt verschiedene Ansätze für ein Messsystem in der Cloud, je nachdem wo die eigentlichen Messungen durchgeführt und Messwerte erhoben und wo diese ausgewertet werden.

A. Reine Auswertung in der Cloud

Neben den klassischen professionellen Messprodukten gibt es bereits seit langer Zeit verschiedene Open-Source-Produkte für Messungen im IP-Bereich. Hierzu zählen z.B. tcpdump [8] oder Wireshark [9]. Mit diesen Programmen ist es leicht und kostengünstig, Messdaten zu erheben [10]. Die Auswertung dieser Daten ist mit den jeweiligen Tools aber zum Teil nur sehr umständlich, lückenhaft oder gar nicht möglich. Hier bieten Messtechnik-Lösungen in der Cloud eine komfortable Möglichkeit, die erfassten Messdaten auszuwerten und entsprechend aufbereiten zu lassen. Die Messdaten werden hierfür lediglich in die Cloud hochgeladen und können online analysiert werden. Die fertige Auswertung wird danach in Form eines Endkundenreports als Download zur Verfügung gestellt. Bei sorgfältig erhobenen Messdaten kann solch ein Ergebnis auf

dem gleichen Niveau einer professionellen Messung durch einen Dienstleister liegen. Je nach Messaufbau können mit dieser Methode internetbasierte und auch lokale Dienste gemessen werden. Abb. 1 und Abb. 2 zeigen einen entsprechenden Messaufbau und verdeutlichen die verschiedenen Datenflüsse. Die Dienst-Daten werden jeweils im Kundennetz an einem Monitoring-Zugangspunkt (z.B. einem sogenannten Test-Access-Point (TAP) oder einem Switch mit Mirror-Port [11]) abgegriffen und zur Auswertung in die QoS-Cloud transferiert, wo anschließend ein Ergebnis generiert wird.

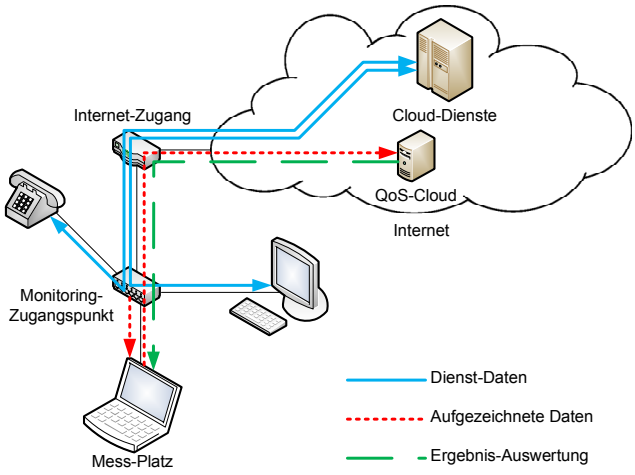


Abb. 1. Vor-Ort-Messung von internetbasierten Diensten mit einer Cloud-basierten Auswertung der Messdaten

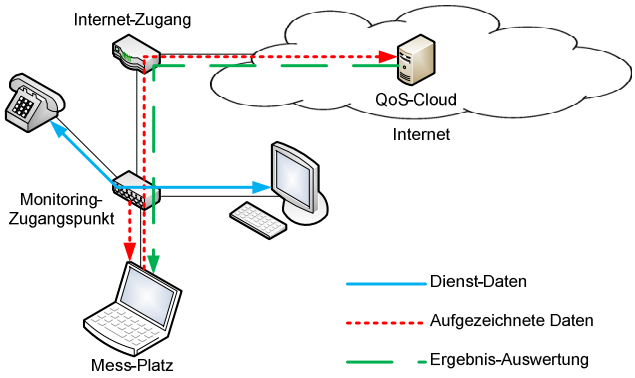


Abb. 2. Vor-Ort-Messung von lokalen Diensten mit einer Cloud-basierten Auswertung der Messdaten

B. *Messungen mit der Cloud*

Eine Erhebung von Messdaten ist mit den erwähnten Tools aber nicht unter allen Umständen möglich. Dies können technische Gründe sein, z.B. wenn die Tools nicht für das zu bewertende System vorhanden sind oder die zu messenden Daten nicht abgegriffen werden können. Zudem können rechtliche Gründe gegen die Nutzung der vorhandenen Dienst-Daten sprechen (z.B. beim Mitschneiden von VoIP-Telefonaten fremder Personen). Diese Einschränkungen können umgangen werden, wenn das Cloud-System durch ein passendes Messtool unterstützt wird. Dies kann z.B. ein kleines Programm sein, das verschiedene Dienste simuliert und gleichzeitig die Messdaten erhebt. Als Gegenstelle werden dann entsprechende Dienste innerhalb des QoS-Messsystems zur Verfügung gestellt.

Abb. 3 zeigt einen Aufbau, bei dem ein regulärer Cloud-Dienst eines Anbieters durch einen eigenen Cloud-Dienst der QoS-Cloud nachgebildet wird. Der Datenfluss zwischen Kundensystem und der Cloud entspricht dabei technisch dem „echten“ Dienst, enthält inhaltlich aber keine sensiblen Daten. Außerdem entfallen eventuell vorhandene Restriktionen des Anbieters, die eine Nutzung des Dienstes durch das Messtool verhindern könnten (z.B. nicht freigegebene Zugangsdaten). Die somit erhobenen Messdaten können dann anschließend automatisch in die Cloud hochgeladen und entsprechend ausgewertet werden.

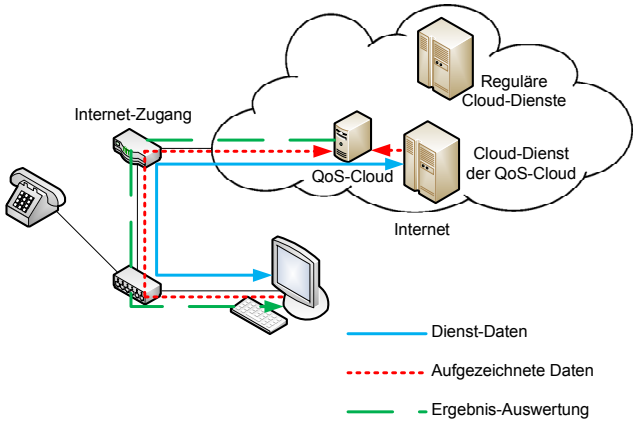


Abb. 3. Vor-Ort-Messung in Kombination mit einer serverseitigen Messung eines internetbasierten Dienstes und einer Cloud-basierten Auswertung der Messdaten

C. *Messungen in der Cloud*

Da die Dienste selbst immer mehr in die Cloud verlagert werden, gibt es hierbei auch weniger real zugängliche Möglichkeiten, um Messdaten zu ermitteln. Hier bietet es sich an, dass direkt in der Cloud Messungen durchgeführt werden. Diese können z.B. an den Übergabepunkten von der Cloud zum End-Nutzer oder zwischen verschiedenen Cloud-Komponenten durchgeführt werden. Ähnlich wie bei der unterstützten Messung durch ein spezielles Messtool können die Daten anschließend automatisch in das Cloud-Messsystem weitergereicht werden. Dies ermöglicht dann eine Echtzeit-Auswertung der Dienste in der Cloud.

Abb. 4 zeigt einen Aufbau, bei dem die Dienst-Daten zwischen verschiedenen Komponenten des Cloud-Dienstes abgegriffen werden. Durch den Einsatz von Virtualisierungstechniken im Bereich des Cloud-Computing stellen die Server und der Monitoring-Zugangspunkt keine physikalischen Komponenten dar, sondern vielmehr logische Komponenten, die flexibel und nach Bedarf konfiguriert werden können.

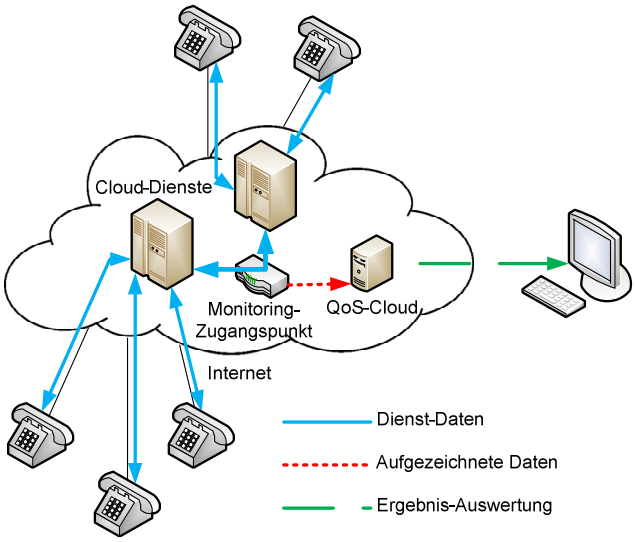


Abb. 4. Serverseitige Messung von Diensten in der Cloud mit einer Cloud-basierten Auswertung der Messdaten

D. Anwendungs-Szenarien

Die verschiedenen Messansätze bieten auch entsprechend unterschiedliche Einsatzmöglichkeiten. Ein großer Vorteil bei der reinen Cloud-basierten Auswertung liegt darin, dass die Messdaten ohne technische Einschränkungen aufgenommen werden können. Der Messaufbau ist dabei unabhängig davon, ob es sich bei den zu messenden Diensten um lokale Dienste aus dem Intranet oder um Cloud-Dienste im Internet handelt. Solange die Daten vor Ort erfasst und aufgezeichnet werden können, ist eine professionelle Auswertung in der Cloud möglich.

Dagegen bieten die Messungen mit und in der Cloud keine Möglichkeit lokale Dienste zu analysieren. Mit diesen Methoden können primär Cloud-basierte Dienste überprüft werden. Messungen mit der Cloud könnten z.B. im Mobilbereich eingesetzt werden, da es auf mobilen Geräten wie Smartphones im Allgemeinen nicht vorgesehen ist, die relevanten Messdaten abzugreifen. Eine entsprechende Anwendung in Form einer Mobile-App könnte den zu prüfenden Dienst nachbilden und hätte dann selbst die Möglichkeit, die erforderlichen Messdaten zu speichern und anschließend von der Cloud auswerten zu lassen.

Da diese Ergebnisse auch die „letzte Meile“ mit einschließen und Alltagssituationen nachbilden, sind so gewonnene Messwerte sowohl für den End-Nutzer als auch für den Dienste-Anbieter interessant. Eine kontinuierliche Überwachung könnte einem Dienste-Anbieter zeigen, wann und unter welchen Umständen es zu Einbrüchen in der Dienstgüte kommt.

Um Engpässe oder Fehler in der Dienst-Infrastruktur selbst zu finden, können entsprechende Messungen „in der Cloud“ eine gute Ergänzung für Dienste-Anbieter sein. Da die Dienste selbst meistens auf verteilten Systemen in einem oder mehreren Rechenzentren verteilt ablaufen, ist eine kontinuierliche Prüfung der entsprechenden Schnittstellen zwischen diesen Einzelkomponenten wichtig.

III. QoS-Messsystem „Testing Cloud“ der Firma Nextragen GmbH

Mit der „Testing Cloud“ [12] der Firma Nextragen GmbH [13] gibt es bereits ein QoS-Messsystem in der Cloud, das nach dem Prinzip der Cloud-basierten Auswertung operiert.

Bei dieser Lösung stehen jedem Interessierten die professionellen Auswertungsalgorithmen der Firma Nextragen GmbH zur Verfügung, um sie direkt selbst nutzen zu können. Im Gegensatz zu langfristig geplanten Dienstleistungen, bei denen der Messspezialist die Messungen vor Ort durchführt, bietet diese Cloud-Variante den Vorteil, dass jederzeit bei Bedarf eine qualitative Bewertung der Netzsituation und der aktuell herrschenden Dienstgüten durchgeführt werden kann.

Bei der Implementierung wurde darauf geachtet, dass die Komponenten des fertigen Produkts nicht nur auf klassisch dedizierten Servern, sondern auch in virtualisierten Umgebungen eingesetzt werden können. Somit kann die zugrunde liegende Technologie zum Einen als „Testing Cloud“ in der „Public Cloud“ als internetweiter Dienst zur Verfügung gestellt werden, zum Anderen aber auch noch in weiteren Produkten als „Private Cloud“ für Intranets oder dediziertes Messtool vertrieben werden.

Des Weiteren sollten bereits etablierte Hilfsmittel und vorhandene Kenntnisse der Netzwerk-Administratoren berücksichtigt werden. Hierzu zählt die Nutzung von Open-Source-Produkten wie tcpdump oder Wireshark, die für einen Netzwerk-Administrator im Allgemeinen bereits zum Alltag gehören. Dies verringert die Einarbeitungszeiten für die Nutzung des Cloud-Dienstes.

Um die Netzsituation zu bewerten, muss der interessierte Nutzer nur einen Mitschnitt der Netzwerkdaten anfertigen, sich bei dem Cloud-Messsystem anmelden und den Mitschnitt in seinem Nutzer-Bereich hochladen. Je nach Größe des Datenmitschnitts steht dann innerhalb von wenigen Minuten das Ergebnis der Netzwerk- und Dienste-Analyse bereit.

Entsprechend der Firmen-Philosophie der Nextragen GmbH werden die Ergebnisse in Form eines leicht verständlichen und repräsentativen Berichts zur Verfügung gestellt. Neben benutzerfreundlichen Einschätzungen der Dienstgüten in Form von „Ampel-Bewertungen“ sind auch die zugrunde liegenden Detailinformationen verfügbar.

Diese Detailinformationen entsprechen der Auswertung, die auch bei einer professionellen Dienstleistung vor Ort durchgeführt werden würde. Somit können erfahrene Netzwerk-Administratoren oder Dienste-Anbieter bzw. Dienste-Nutzer ohne großen zeitlichen und finanziellen Aufwand prüfen, wie sich die Netzsituation entwickelt.

Durch die abnehmende Distanz zwischen IP-basierten Diensten und dem End-Nutzer wird der Bedarf an qualitativen Aussagen aber auch im semi-professionellen und privaten Umfeld immer größer.

Der Wandel vom klassischen ISDN- oder Analog-Telefonanschluss zu IP-basierten VoIP-Anschlüssen und die Verbreitung von IPTV führen dazu, dass immer mehr Privat-Haushalte direkten Kontakt mit Netzwerk-Infrastrukturen

haben. Und da die angebotenen Dienste wie Video und Telefonie zur Gruppe der Echtzeit-Dienste gehören, werden an diese Infrastruktur entsprechend hohe Anforderungen gestellt.

Mit Hilfe von Cloud-basierten QoS-Messsystemen wie der „Testing Cloud“ haben auch End-Nutzer die Möglichkeit, ihre Netze zu testen und die vorhandenen Dienstgüten bewerten zu lassen. Die Messerfahrung der Nextragen GmbH im Bereich der Triple-Play-Dienste ermöglicht eine automatische Klassifizierung der erhobenen Messdaten. So können die einzelnen Dienste in Form von „Ampel-Bewertungen“ mit den einfach verständlichen Prädikaten „Gut“, „Mittelmäßig“ und „Schlecht“ versehen werden.

Diese Ergebnisse bieten auch dem unerfahrenen Nutzer die Möglichkeit, auftretende Probleme genauer erfassen zu können. Mit Hilfe der Ergebnis-Details kann ein Dienste-Anbieter bei Bedarf schneller mögliche Ursachen finden und entsprechende Gegenmaßnahmen anbieten und einleiten.

A. *Aufbau und Eigenschaften*

Die „Testing Cloud“ der Firma Nextragen GmbH basiert auf dem hauseigenen Analyse-Tool „TraceMon“ [14]. Hierbei handelt es sich um eine passive Analyse-Software, die für Netzwerk-Langzeit-Messungen entwickelt wurde und auf die Analyse und Bewertung von Triple-Play-Diensten spezialisiert ist.

Diese Software kann im Zuge von Dienstleistungen der Firma Nextragen GmbH für den kurzzeitigen Einsatz gebucht oder für die häufige und wiederholte Nutzung erworben werden.

Die „Testing Cloud“ ist durch die Umsetzung als Cloud-Dienst im Vergleich kostengünstiger und kann kurzfristig für Einzeltests genutzt werden, bietet aber dennoch dieselben Ergebnisse wie die professionelle Messsoftware.

Um die „Testing Cloud“ nutzen zu können, muss sich der interessierte Nutzer nur auf der Website anmelden und kann direkt seine Messdaten hochladen. Diese Daten können im weit verbreiteten pcap-Format oder im eth-Format vorliegen. Dies ermöglicht die einfache Messung mit Hilfe vorhandener Tools wie tcpdump oder Wireshark (pcap-Format) und die direkte Messung im End-Nutzer-Bereich auf Fritz!Box-Internet-Routern der Firma AVM (eth-Format) [15].

Nach dem Hochladen der Messdaten werden diese auf den Servern der Firma Nextragen GmbH analysiert und ausgewertet. Das Ergebnis wird dann in Form eines PDF-Berichts zur Verfügung gestellt und kann vom Nutzer heruntergeladen werden (vgl. Abb. 5).

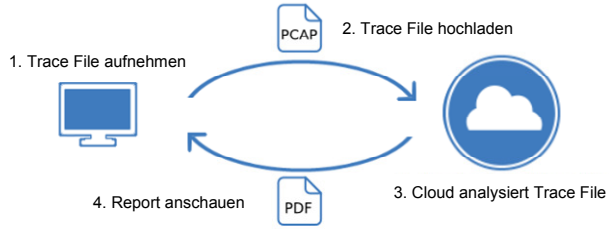


Abb. 5. Ablaufschema der "Testing Cloud" aus der End-Nutzer-Beschreibung

Technisch werden die hochgeladenen Messdaten dafür in eine Abarbeitungs-Queue eingereiht und nacheinander vom Analyse-Core verarbeitet (vgl. Abb. 6). Dieser führt zuerst eine Daten-Analyse durch, um nicht relevante Daten für die Triple-Play-Analyse herauszufiltern. Die verbleibenden Daten werden dann nach anerkannten Messmethoden bewertet und die Ergebnisse entsprechend für einen Report aufbereitet. Zusätzlich zu diesen Detail-Informationen kommen Schwellenwert-basierte Klassifizierungen der verschiedenen Werte. Diese Einteilung ermöglicht es, „Ampel-Bewertungen“ zu generieren, die auch komplexe Mess-Ergebnisse auf einfache und verständliche Art und Weise darstellen können.

Abb. 6 zeigt den technischen Ablauf vom Hochladen der Daten bis zum Herunterladen des Ergebnisses:

1. Einreihen der hochgeladenen Daten in eine Warteschlange zur Abarbeitung.
2. Filtern der Daten, damit nur relevante Informationen analysiert werden.
3. Analyse der gesammelten Informationen nach anerkannten Standards (z.B. G.107 [16], RFC3550 [17], VS-/VT-Modell [18][19]).
4. Bewertung und Aufbereitung der Ergebnisse, um „Ampel-Ergebnisse“ zu erhalten.
5. Erzeugung eines PDF-Reports und Bereitstellen zum Download.

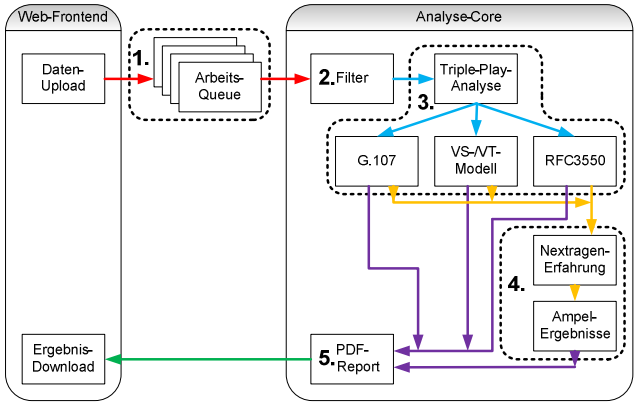


Abb. 6. Technischer Ablauf in der "Testing Cloud"

B. *Testmöglichkeiten in Beispielen*

Da die „Testing Cloud“ nach dem Prinzip der Cloud-basierten Auswertung arbeitet, können sowohl Internet-Dienste als auch lokale Dienste analysiert werden. Als Voraussetzung muss es lediglich möglich sein, die zu untersuchenden Daten lokal erfassen zu können. Dies kann im professionellen Bereich z.B. durch den Einsatz von TAPs oder von Switches mit Spiegel-Ports erreicht werden. Da beide Varianten flexibel und an verschiedenen Stellen im Netzwerk platziert werden können, bieten sie eine Möglichkeit zum gezielten Messen an kritischen Systempunkten (vgl. Abb. 7). Im privaten Umfeld gibt es bei einigen Internet-Routern die Möglichkeit, den Internet-Datenverkehr direkt aufzuzeichnen. Die fehlende Flexibilität bzgl. des Messortes kann häufig vernachlässigt werden, da

im Heimbereich meistens keine größeren Netzwerke mit weiteren Netzkomponenten vorhanden sind. Abb. 8 zeigt beispielhaft solch einen vereinfachten Aufbau eines Heimnetzes mit einer Fritz!Box und den angeschlossenen Geräten.

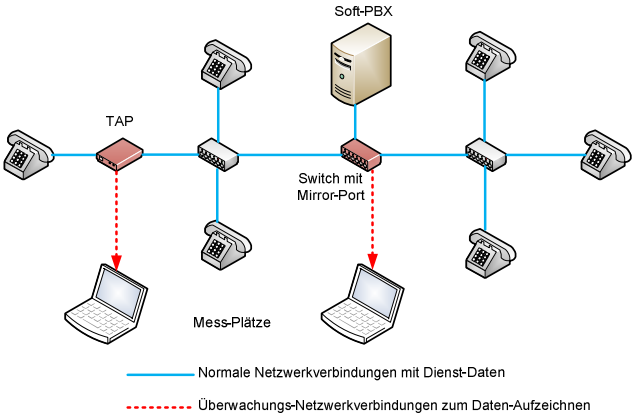


Abb. 7. Flexible Daten-Erfassung innerhalb eines größeren Netzwerks mit einem TAP oder Mirror-Port

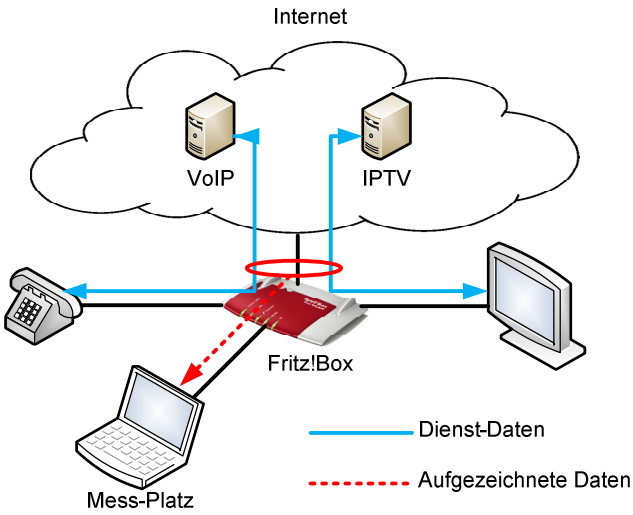


Abb. 8. Daten-Erfassung mit einer AVM Fritz!Box

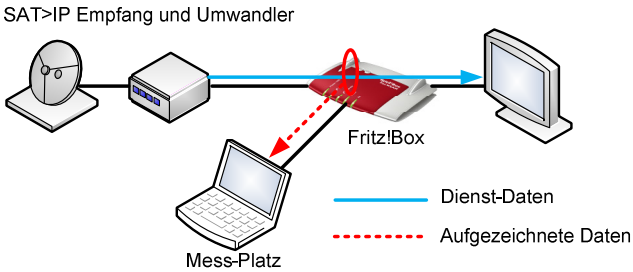


Abb. 9. Daten-Erfassung bei SAT>IP innerhalb eines rudimentären Heimnetzwerkes

Beide Varianten bieten eine gute Möglichkeit, z.B. VoIP-Telefonie auszuwerten und im Fehlerfall eine erste Ursachenanalyse durchzuführen. Zusätzlich zu diesem klassischen Einsatzbereich der Telefonie kann aber auch der zunehmende Bereich des IP-basierten Fernsehens mit untersucht werden. Hierzu zählen nicht nur IPTV-Angebote aus dem Internet, sondern auch die zunehmende hausinterne IPTV-Übertragung von klassisch empfangenen Fernseh-Angeboten. Sowohl für Kabel- (DVB-C) als auch SAT-Empfang (DVB-S/S2) gibt es entsprechende Umwandler [20], die nach dem SAT>IP Standard [21] einen privaten IPTV-Stream zur Verfügung stellen. Abb. 9 zeigt einen einfachen Aufbau mit einer Satelliten-Anlage und einem Fernseher. In diesem Fall kann auch ohne Bedenken an einem zentralen Router gemessen werden. In größeren Haushalten mit mehreren Fernsehern, die über ein ausgedehnteres Netzwerk miteinander verbunden sind, empfiehlt sich dagegen der Einsatz der professionelleren TAPs oder von Mirror-Ports an den vorhandenen Switches.

IV. QoS Measurement Management Systeme (MMS) im Allgemeinen

Die „Testing Cloud“ zeigt, dass es neben den verschiedenen technischen Ansätzen zur Realisierung der eigentlichen Messung bei einem Cloud-basierten QoS-Messsystem auch verschiedene Ansätze bei der Ergebnis-Auswertung gibt.

Während klassische Messsysteme meistens sehr detaillierte Ergebnisse liefern, die von einem geschulten Personal oder Dienstleister ausgewertet werden müssen, können Cloud-basierte Systeme auch leicht verständliche Zusammenfassungen und Bewertungen liefern. Zudem können diese Fähigkeiten kontinuierlich vom Anbieter der Cloud-Lösung erweitert und verbessert werden.

A. *Anforderungen an QoS-MMS*

Als Grundlage für solch ein QoS-Messsystem müssen verschiedene Anforderungen erfüllt werden. Hierzu zählen nicht nur technische Herausforderungen, sondern auch rechtliche Rahmenbedingungen.

1) *Zuverlässigkeit der Messwerte*

Die Basis jeder Messung sind die Messdaten. Dies gilt unabhängig davon, ob es sich um klassische Messtechnik handelt oder um eine Cloud-basierte Messlösung. Zuverlässige Messwerte sind immer eine zwingende Voraussetzung für aussagekräftige Ergebnisse. Während bei der klassischen Messtechnik spezielle Messsysteme zum Einsatz kommen und geschultes Personal für eine korrekte Durchführung der Messung sorgt, ist dies bei einer Cloud-Lösung nicht zwangsläufig gegeben.

Bei der reinen Auswertung in der Cloud kann nicht überprüft werden, in welcher Art und Weise die Messdaten erhoben wurden und ob der Aufbau messtechnischen Ansprüchen genügt. Für ungeübte Nutzer sollten deshalb ausführliche Anleitungen und Hinweise vorhanden sein, um eine sinnvolle Messung zu ermöglichen.

Falls sich diese passiven Hilfsmittel als nicht ausreichend herausstellen, können aktive Hilfsmittel wie

automatisierte Aufzeichnungstools helfen. Wenn Messungen mit der Cloud durchgeführt werden, verringert sich die Anzahl möglicher Variablen. Das bereitgestellte Tool kann z.B. sicherstellen, dass das genutzte Aufzeichnungsgerät überhaupt in der Lage ist, korrekte Messwerte zu liefern. Andernfalls können z.B. Hinweise auf zu geringe Systemleistung oder fehlende Hardwarevoraussetzungen gegeben werden.

Steht nicht der Dienst selbst im Fokus, sondern soll analysiert werden, wie sich die zugrunde liegende Infrastruktur auf die Dienstgüte auswirkt, empfiehlt es sich, einen eigenen Dienst des Cloud-Messsystem-Anbieters als Referenzgegenstelle einzusetzen. In diesem Fall kann sichergestellt werden, dass eine vorhandene Infrastruktur nicht fälschlicherweise bemängelt wird, nur weil ein unzureichender Dienst als Grundlage genutzt wurde.

Bei Messungen in der Cloud können die Unsicherheiten noch weiter minimiert werden, wenn das Messsystem zusammen mit dem Dienste-Anbieter eingerichtet wird. Durch die Integration in den eigentlichen Cloud-Dienst ähnelt diese Umsetzung einer klassischen Integration von Messgeräten in die Unternehmens-Infrastruktur. Unterschiede ergeben sich allerdings durch die flexible Gestaltung heutiger Rechenzentren. Mit Hilfe neuer Technologien wie z.B. Software-Defined-Networking kann das Messsystem kurzfristig an die Bedürfnisse des Dienste-Anbieters angepasst werden [22]. Hierbei ist aber darauf zu achten, dass sich nicht alle modernen Technologien vorbehaltslos für ein QoS-Messsystem eignen. So kann z.B. die Nutzung von Virtualisierung zu Verfälschung von Echtzeit-Messwerten führen [23].

2) Ort der Messwertaufnahme

Aus den Überlegungen zur Zuverlässigkeit der Messwerte ergeben sich zwei Ansätze für mögliche Messorte. Im Falle einer reinen Cloud-basierten Auswertung für lokale Intranet-Dienste sollten ausreichend Informationen und Anleitungen angeboten werden. So hat auch ein ungeübter Administrator die Möglichkeit, den besten Ort für seine Messungen zu finden.

Um internetbasierte Cloud-Dienste zu messen und zu überwachen, empfiehlt sich eine Kombination aus Messungen mit der Cloud und Messungen in der Cloud. Durch die verteilte Anordnung der einzelnen Messknoten können mehrere Transitstationen vom Dienste-Anbieter bis hin zum End-Nutzer mit einbezogen werden.

3) Datenschutz

Das Erfassen von Daten ist mit heutigen Mitteln ohne großen Aufwand möglich. Die Herausforderung liegt in der datenschutz-konformen Verarbeitung dieser Daten. Im Fall der selbst ermittelten Messdaten im lokalen Firmenumfeld kann nur ausdrücklich auf die geltenden Regeln (u.a. die gesetzlichen Bestimmungen) und die möglichen Konsequenzen bei einer Missachtung hingewiesen werden. Aus diesem Grund haben Cloud-Angebote wie die „Testing Cloud“ entsprechende AGBs und Nutzungsbedingungen [24]. Weitere technische Schutzmaßnahmen können hier prinzipbedingt nicht umgesetzt werden, da sie nur die Aussagefähigkeit des Ergebnisses einschränken würden (z.B. Ausblenden der IP-Adressen).

Wenn Messungen mit oder in der Cloud durchgeführt werden, können dagegen Vorkehrungen getroffen werden, so dass keine „Fremddaten“ erfasst werden. Da bekannt ist, welches System die Messungen durchführt bzw. welche Cloud-Komponenten miteinander interagieren, können andere Daten herausgefiltert werden. Somit bleiben im Allgemeinen nur noch die personenbezogenen Daten des eigentlichen Nutzers über. Mit entsprechenden AGBs und Nutzungsbedingungen ist es möglich, diese Daten aufzubereiten und als Ergebnis zur Verfügung zu stellen. Sollen die erhobenen Daten nicht nur dem Nutzer des Dienstes, sondern auch dem Anbieter zur Verfügung gestellt werden, ist es hierbei aber wichtig zu beachten, dass eine Weiterverarbeitung nur anonymisiert erfolgen darf [25]. Zusätzlich sollten fertig ausgewertete Rohdaten gelöscht werden.

4) Bereitstellung der Ergebnisse

Eine weitere wesentliche Herausforderung ist die Aufbereitung und Bereitstellung der Ergebnisse. Da heutige Systeme eine Vielzahl von Messparametern erfassen können, ist eine sinnvolle Auswertung eine Hauptaufgabe moderner Messsysteme. Wie das Beispiel der „Testing Cloud“ zeigt, liegt hier ein wesentlicher Anteil an dem Know-How und der Erfahrung des Messsystem-Herstellers. Je nach Ausführung des Systems können die Ergebnisse individuell als Report zur Verfügung gestellt werden oder auch als Zusammenfassung in übergeordnete Überwachungssysteme des Dienste-Anbieters eingespeist werden. Die konkrete Umsetzung hängt dabei von den Anforderungen und Wünschen des Auftraggebers ab.

B. Konzept für ein QoS-MMS für Triple Play Services

Die „Testing Cloud“ der Firma Nextragen GmbH stellt eine Umsetzung eines QoS-MMS für beliebige fremde (d.h. nicht Nextragen-eigene) Dienste dar. Es können wiederholt Messungen durchgeführt werden, die dann aber manuell hochgeladen werden müssen, bevor sie als Ergebnis im System verwaltet werden können.

Für ein automatisches Measurement Management System eignet sich besser eine Kombination aus den Ansätzen für Messungen mit und in der Cloud. Analog zu den Forschungsprojekten der EU und dem LMAP der IETF sollten hierbei mehrere Messknoten an systemkritischen Übergabepunkten verteilt werden. Diese können die angebotenen Dienste durch passive Messmethoden bereits im Kernbereich kontinuierlich überwachen. Zusätzlich können die Dienstgüten bis zum End-Nutzer gemessen werden, wenn dieser ein entsprechendes Hilfstool einsetzt, mit dem aktive Zusatzmessungen durchgeführt werden können. Im Consumer-Bereich könnte dies z.B. ein normales Computer-Programm oder eine mobile Smartphone-App sein, um auch die letzte Meile zu überwachen. Bei Enterprise-Kunden könnte diese Komponente aber auch direkt mit in die Teilnehmer-Endgeräte (Customer Premises

Equipment, CPE) integriert werden, die der Anbieter für die Nutzung seiner Dienste zur Verfügung stellt [26]. Abb. 10 zeigt exemplarisch einen möglichen Aufbau eines solch verteilten Systems.

Je nach gewünschter Nutzung des Systems können die anonymisierten und kumulierten Daten im Überwachungssystem des Dienste-Anbieters selbst bzw. beim Betreiber der Infrastruktur (in der Regel dann der Internet Service Provider) zur Verfügung gestellt werden. Als zusätzlichen Service können die individuellen Ergebnisse auch den jeweiligen End-Kunden zugestellt werden und als Beleg über die erhaltenen Leistungen genutzt werden.

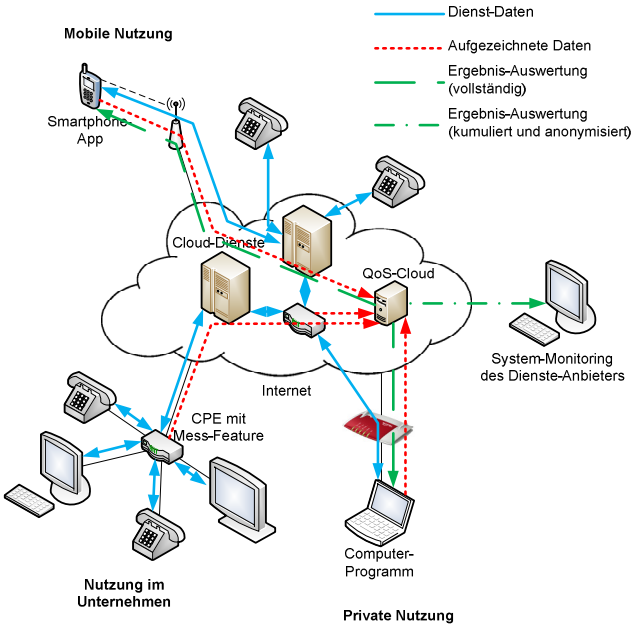


Abb. 10. Mögliche Anordnung der verschiedenen Messknoten in einem QoS-MMS

V. Zusammenfassung mit Ausblick

Im Rahmen dieser Arbeit wurden verschiedene Ansätze für die Realisierung eines QoS-Messsystems in einer Cloud-Umgebung vorgestellt, die jeweils für unterschiedliche Einsatzszenarien geeignet sind. Um praxisnah zu bleiben, wurde das Thema am Beispiel der Produkte der Firma Nextragen GmbH erörtert. Die Firma Nextragen GmbH bietet mit ihrer „Testing Cloud“ bereits eine Lösung an, mit der unabhängig und herstellerübergreifend verschiedene Dienste bewertet werden können. Auch andere Firmen, wie z.B. CapAnalysis [27], Paessler [28] und CloudShark [29] haben diese neue, kostengünstige Messmöglichkeit für Netzwerkanalysen und zur Feststellung der Dienstgüten in einer Cloud-Umgebung „entdeckt“. Dies bestätigt die Wichtigkeit der im Rahmen dieser Arbeit vorgestellten Problematik.

Um eine automatisierte und kontinuierliche Vermessung größerer Systeme zu ermöglichen, werden aber auch entsprechend größere Messsysteme benötigt. An allgemeinen Grundlagen für solche Systeme wird zwar auch in den EU-Projekten geforscht, um ein System wie das vorgestellte QoS-MMS zu realisieren, werden aber noch spezialisierte Komponenten für Triple Play Services benötigt. Entsprechende weitergehende Forschungen und Implementierungen von Prototypen stehen bereits an. Es lohnt sich, in diese Richtung weitere Arbeiten durchzuführen.

VI. Quellen

[1] http://www.itu.int/rec/T-REC-Y.2001, *Definition of NGN,* page last viewed July 2015

[2] DIRECTIVE 2009/140/EC OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 25 November 2009 amending Directives 2002/21/EC (Official Journal EU L.337/37)

[3] DIRECTIVE 2009/136/EC OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 25 November 2009 amending Directive 2002/22/EC (Official Journal EU L 337/11)

[4] http://www.leone-project.eu, *leone – From global measurements to local management*, page last viewed July 2015

[5] http://www.ict-mplane.eu, *mPlane – an Intelligent Measurement Plane for Future Network and Application Management*, page last viewed July 2015

[6] https://tools.ietf.org/wg/lmap, *Large-Scale Measurement of Broadband Performance*, page last viewed July 2015

[7] Daniel Felsmann, *Cloud Computing – Basistechnologie, Architektur, Erfolgsfaktoren, Herausforderungen und die aktuelle Marktsituation,* GRIN Verlag 2010

[8] http://www.tcpdump.org, *tcpdump & libpcap*, page last viewed July 2015

[9] http://www.wireshark.org, *Wireshark*, page last viewed July 2015

[10] Thomas Kaufmann, *Netzwerk-Diagnose – Auf Päckchenfang; Paketanalyse mit Ethereal*, iX 5/2005, S. 153

[11] Mark Ciampa, *Security+ Guide to Network Security Fundamentals,* Cengage Learning, 2011

[12] http://www.testing-cloud.com, *Testing Cloud*, page last viewed July 2015

[13] http://www.nextragen.de, *Nextragen*, page last viewed July 2015

[14] http://www.nextragen.de/produkte/tracemon, *TraceMon*, page last viewed July 2015

[15] http://praxistipps.chip.de/fritzbox-datenverkehr-mitschneiden_9989, *Fritzbox: Datenverkehr mitschneiden*, page last viewed July 2015

[16] https://www.itu.int/rec/T-REC-G.107, *G.107 – The E-model: a computational model for use in transmission planning*, page last viewed July 2015

[17] https://www.ietf.org/rfc/rfc3550.txt, *RTP: A Transport Protocol for Real-Time Applications*, page last viewed July 2015

[18] S. Paulsen and T. Uhl, *The new, parameterized VS Model for determining the quality of video streams in the video-telephony Service*, J. Telecom. Review. 8–9, 1155–1166 (2012).

[19] S. Paulsen and T. Uhl, *The new, parameterized VT Model for determining the quality in the video-telephony service*, Bulletin of the Polish Acadamy of Sciences Technical Sciences, Vol. 62, No. 3, 2014 DOI: 10.2478/bpasts-2014-0045.

[20] http://www.digitaldevices.de/Octopus_Net.html, *Octopus Net HDTV Netzwerktuner*, page last viewed July 2015

[21] http://www.beuth.de/de/norm/din-en-50585-vde-0855-585-2015-02/207271832, *DIN EN 50585:2015-02; VDE 0855-585:2015-02 Communications protocol to transport satellite delivered signals over IP networks*, page last viewed July 2015

[22] Gary Lee, *Cloud Networking: Understanding Cloud-based Data Center Networks,* Morgan Kaufmann, 2014

[23] Miroslav Voznak, Jiri Slachta, Jan Rozhon, *Real-Time Applications on Virtualized Machines,* Mathematical Methods and Computational Techniques in Electrical Engineering (MMACTEE '11)

[24] http://nextragen.dynalias.org:8000/tos, *Terms of Service*, page last viewed July 2015

[25] G. Recht, *Telemediengesetz – Tmg,* Createspace Independent Pub, 2014

[26] http://www.ict-mplane.eu/sites/default/files/public/public-page/public-deliverables/988mplane-d14.pdf, *mPlane – Final Requirements, Architecture, Access Control & Data Protection, Interface Specifications*, page last viewed July 2015

[27] http://www.capanalysis.net, *CapAnalysis*, page last viewed July 2015

[28] http://www.paessler.com, *Paessler*, page last viewed July 2015

[29] https://appliance.cloudshark.org, *CloudShark Applicance*, page last viewed July 2015

Das C++ Actor Framework im Leistungsvergleich

Marian Triebe, Dominik Charousset, Raphael Hiesgen, Thomas C. Schmidt
Internet Technologies Group, Dept. Informatik, HAW Hamburg, Germany
{marian.triebe, dominik.charousset, raphael.hiesgen, t.schmidt}@haw-hamburg.de

***Zusammenfassung*—Der langjährige Trend in der CPU-Entwicklung vermehrt Kerne und nicht die Leistung pro Kern. Die dabei entstehenden Multicore-Architekturen sind nur mittels nebenläufiger Programmierung gleichmäßig auslastbar. Gleichzeitig lassen die Omnipräsenz des Internets zwischen heterogenen, weltweit verteilten Komponenten, insbesondere aber spezialisierte Internet-Lösungen wie Clouds, das Internet der Dinge (IoT) etc. verteilte Anwendungen immer stärker in den Vordergrund treten. Ein Lösungskonzept für die transparente, robuste Programmierung paralleler und verteilter Systeme ist das Aktorenmodell. Das C++ Actor Framework (CAF) ist unser Beitrag zur Realisierung dieses Konzepts. CAF kann sowohl hochparallele Maschinen, als auch heterogene Hardware-Komponenten wie GPUs, aber auch lose gekoppelte Systeme im Internet und stark beschränkte IoT-Knoten in einem Programmsystem verknüpfen.**

In diesem Beitrag stellen wir die Leistungsmerkmale von CAF vor. Wir vergleichen CAF zunächst mit dem elementaren Messaging Ansatz von MPI, wobei vor allem die Skalierbarkeit von Interesse ist. Wir stellen weiterhin CAF anderen Implementierungen des Aktorenmodells gegenüber und vergleichen Speicherverbrauch und Terminierungszeit. Unsere Messungen zeigen, dass CAF in den gemessenen Kategorien durch hochskalierbares Design und effiziente Implementierung in C++ überlegene Leistungswerte aufweist. Außerdem präsentieren wir Messungen für die Bereiche der eingebetteten und mobilen Systeme, wo effiziente RAM Nutzung eine wichtige Rolle spielt.

***Index Terms*—Actor Model, C++, Message-oriented Middleware, Performance Measurement**

I. Einleitung

Die Software-Industrie sieht sich derzeit gleich mehreren Trends gegenüber. Der größte Umbruch war das Ende des stetigen Anstiegs der Taktfrequenzen um die Jahrtausendwende und der Beginn der "Multicore-Ära". Die Notwendigkeit der nebenläufigen Ausführung, um von neuen Hardware-Generationen profitieren zu können, erforderte ein Umdenken bei der Software-Entwicklung. Gleichzeitig verbindet die Cloud mehr und mehr Bereiche des täglichen Lebens, was auch bedeutet, dass immer mehr Computer-Programme kommunizieren und in einem globalen Netz organisiert sind. Der dritte Trend ist die Automatisierung der Maschinen-Kommunikation und die Schaffung des IoT. Hierbei lösen viele Kleinstrechner im Verbund Probleme, was ein hohes Maß an Organisation und Verlässlichkeit erfordert. Heterogene Ablaufumgebungen bilden den vierten Trend. Technologien wie OpenCL [1] erlauben das Ausführen beliebiger Programme auf der Grafikkarte, wodurch Lösungen für stark parallelisierbare Probleme, wie beispielsweise Kryptographie oder Video(de)kodierung, enorme Geschwindigkeitssteigerung erreichen können.

Das Aktorenmodell [2] beschreibt Computer-Programme als Summe einzelner Software-Komponenten, die durch den Austausch von Nachrichten kommunizieren. Da Aktoren voneinander unabhängig agieren, kann eine intelligente Laufzeitumgebung nebenläufige Aktoren beliebig auf einem Rechner und auch über mehrere Rechner hinweg parallel ausführen. Dabei ist der logische Programmaufbau unabhängig von der realen Verteilung. Darüber hinaus bietet das Aktorenmodell ein starkes Fehlermodell auf Basis von uni- und bidirektionaler Überwachung mit nachrichtenbasierter Fehlersignalisierung. Beim Ausfall einer oder mehrerer Aktoren lassen sich Teile einer Anwendung im laufenden Betrieb neu verteilen, wodurch eine Anwendung tolerant gegenüber ausfallenden Systemkomponenten wird.

Konzeptionell eignet sich das Aktorenmodell zur Entwicklung nebenläufiger und verteilter Systeme gleichermaßen. Auf Mehrkernsystemen skalieren Aktorensysteme, indem ein Problem in viele Teilprobleme zerlegt wird und entsprechend viele Aktoren gestartet werden. Dadurch entstehen im Vergleich zur Anzahl physisch vorhandener CPU-Kerne viele Aktoren, die jedoch einzeln eine geringe Komplexität haben. Gemäß dem Amdahlschen Gesetz [3] minimiert dieser Ansatz die sequentielle Komponente eines Programms und ermöglicht effizientes Skalieren mit der Anzahl vorhandener Kerne.

Mit dem C++ Actor Framework (CAF) [4] stellen wir unser Konzept zur transparenten und elastischen Programmierung von Aktoren in C++ vor und vermessen seine Leistungsmerkmale. Hierbei analysieren wir einerseits die Zusatzkosten des CAF-Laufzeitsystems, andererseits vergleichen wir die Performance in markanten Benchmarks mit den bekanntesten Aktor-Laufzeitsystemen sowie mit MPI. Wir konzentrieren uns insbesondere auf Aspekte der Skalierbarkeit und können für CAF ein hervorragendes Leistungsverhalten zeigen.

Die Architektur von CAF sowie die Hauptkomponenten für eine effiziente Laufzeit werden in Abschnitt II vorgestellt. In Abschnitt III vergleichen wir die Leistungsmerkmale von CAF mit anderen Aktor-Laufzeitsystemen. Eine Leistungsmessung für verteilte und heterogene Systemen findet sich in Abschnitt IV. Abschließend möchten wir unsere Ergebnisse in Abschnitt V diskutieren.

II. Das C++ Actor Framework und verwandte Arbeiten

Mit stagnierenden Taktraten und dem absehbaren Ende des Mooreschen Gesetzes [5] stieg auch das Interesse innerhalb der Entwicklergemeinde an nativen Programmiersprachen, um auf ein größeres Optimierungspotential zurückgreifen zu

können. Insbesondere C++ erlebte durch diese Entwicklung eine Renaissance, was sich unter anderem im Erscheinen von zwei neuen Standard-Versionen (2011 und 2014) widerspiegelt. Jedoch bleiben die verfügbaren Programmierwerkzeuge zur Entwicklung nebenläufiger Anwendungen vergleichsweise niedrigstehend und beschränken sich auf Synchronisationsprimitive wie Mutexe und Condition-Variablen, welche zur korrekten Benutzung viel Expertenwissen erfordern [6]. Außerdem kann es bei naivem Speicherlayout zu False Sharing kommen, was zu langsamerer Laufzeit trotz einem Ansatz mit geringerer Abstraktion führt [7]. Ähnlich niedrigstehende Primitive stellt C++ zur verteilten Programmierung bereit (Programmierung auf Socket-Ebene). Dies ist fehleranfällig, komplex und aufwendig.

Das Aktorenmodell adressiert sowohl Nebenläufigkeit als auch Verteilung auf einer einheitlichen Abstraktionsschicht. Mit CAF möchten wir auf Basis des Aktorenmodells eine hochstehende Programmierschnittstelle für Entwickler anbieten, die Nebenläufigkeit sowie Verteilung in einem robusten Programmierparadigma zusammenfasst. Ziel ist dabei, das Aktorenmodell für den Einsatz in Performance-kritischer Infrastruktur-Software, im Embedded-Bereich und für das IoT anzupassen und eine Laufzeitumgebung anzubieten, die Ressourcen effizient verwaltet und ein effizientes Scheduling vornimmt. Mit Ausrichtung auf diese Anwendungsfelder ergeben sich Anforderungen an Scheduling sowie Ressourcen-Management, die beim Entwurf bisheriger Implementierungen nur eine untergeordnete Rolle hatten. Traditionelle Implementierungen des Aktorenmodells basierten meist auf virtualisierten Umgebungen wie der JVM [8] und verfolgten entsprechend andere Ziele. Eine Implementierung für C++, die sich zumindest am Aktorenmodell orientiert, ist Charm++ [9], das konzeptionell jedoch auf MPI aufsetzt und seinen Fokus auf die Entwicklung von Anwendungen für Cluster und Supercomputer legt.

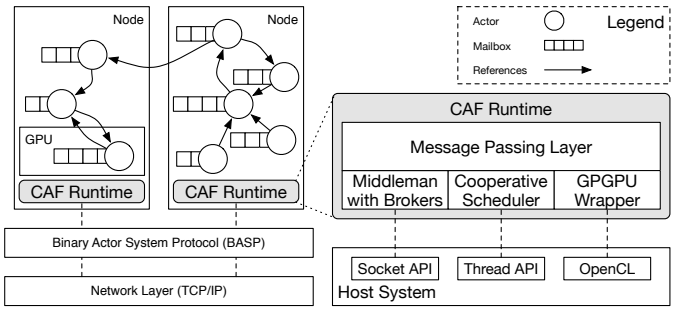


Abbildung 1. CAF-Architektur

Aktoren in CAF laufen in einer Laufzeitumgebung, die Nachrichtenversand, lokales Scheduling sowie Queue-Management zur Verfügung stellt.

Die Kommunikation mit entfernten Aktoren wird durch diese Laufzeitumgebung transparent bereitgestellt. Logische Verbindungen zwischen Aktoren werden durch einen "Middleman" hergestellt, der als Multiplexer dient. Auf dem Hostsystem wird daher nur ein offener Port benötigt. Abbildung 1 zeigt die Architektur einer verteilten CAF-Anwendung. Aktoren sind sich der physischen Verteilung nicht bewusst, sondern formen ein logisches Kommunikationsnetzwerk, welches von der Laufzeitumgebung prinzipiell beliebig auf physische Knoten verteilt werden kann. Diese flexible Topologie wird ermöglicht durch das "Binary Actor System Protocol" (BASP). Verteilte Laufzeitumgebungen realisieren ihre Kommunikation über Middleman-Instanzen. Die Hauptaufgabe eines Middlemans ist es, von der Socket-API des Hostsystems zu abstrahieren und eine nachrichtenbasierte Schnittstelle für Netzwerkkommunikation bereitzustellen. Paket- und Byte-Streams auf der Netzwerkschicht werden als Nachrichten sogenannten Brokern zugestellt. Broker sind Aktoren, die asynchrones I/O ausführen und in der Event-Schleife des Middlemans laufen. Entfernte Aktoren werden durch den "BASP Broker" kontaktiert, welcher interne Aktor-Nachrichten an das Netzwerk weiterleitet. Ein kooperativer Scheduler organisiert paralleles sowie faires Ausführen von Aktoren auf einem lokalen Knoten und verwendet die Threading-API aus der C++ Standard-Bibliothek. Der GPGPU-Wrapper verbirgt heterogene Hardware-Komponenten hinter einer Fassade. Er erstellt Aktoren, die Aufgaben an OpenCL-Kernel weiterleiten. Nachrichten werden zur Laufzeit in OpenCL-kompatible Datentypen gewandelt und umgekehrt.

In den folgenden Abschnitten möchten wir zunächst zwei Algorithmen vorstellen, die für einen effizienten Nachrichtenaustausch essentiell sind. Nachfolgend wird zudem der Scheduler im Detail vorgestellt, da dieser maßgeblich für die Leistungsmerkmale von CAF auf Mehrkernsystemen verantwortlich ist.

A. Mailbox

Die Mailbox-Implementierung ist eine der wichtigsten Komponenten in einem Aktor-Laufzeitsystem. Empfängt ein Aktor von mehreren Sendern gleichzeitig Nachrichten, so kann die Mailbox zu einem Flaschenhals werden und somit die gesamte Systemperformance negativ beeinflussen. Eine hohe Performance bei nebenläufiger Ausführung ist nur möglich, wenn parallele Lese- und Schreibzugriffe keinen signifikanten Synchronisations-Aufwand verursachen. Die Wahl des richtigen Algorithmus ist daher entscheidend. Die Mailbox in CAF ist eine "Single-Reader-Many-Writer-Queue". Diese Implementierung erlaubt paralleles Schreiben in die Mailbox, wobei nur der Besitzer der Queue Nachrichten aus dieser Mailbox entnehmen kann. Unsere Implementierung basiert auf einem nicht-blockierenden Stack (LIFO), für den alle Zugriffe mit Hilfe einer einzigen atomaren compare-and-swap (CAS)-Operation realisiert sind. Um die Nachrichten in der korrekten Reihenfolge auszulesen ist dieser Stack mit einem internen Cache kombiniert, der Nachrichten in FIFO-Reihenfolge übersetzt und nur dem Besitzer zugänglich ist. Die Implementierung der Mailbox leidet nicht unter dem ABA-Problem bei parallelem Zugriff in CAS-basierten Systemen [10], da durch die Entkopplung über den internen Cache jedes Element maximal einem Thread zur Zeit zugänglich ist. Die Komplexität der Mailbox liegt bei $O(1)$ für Einfüge-

Operationen. Bei der Entnahme liegt der Durchschnitt bei $O(1)$, jedoch im schlechtesten Fall durch das umsortieren der Nachrichten in den internen Cache bei $O(n)$, wobei n die Anzahl der Nachrichten ist. Eine detailliertere Veranschaulichung der Mailbox wurde in vorherigen Publikationen gezeigt [11].

B. Copy-On-Write Nachrichten

Die Implementierung der Nachrichtenschicht von CAF verwendet Tupel mit Wertesemantik. Jedoch kann ein Tupel zwischen mehreren Aktoren implizit geteilt sein, so lange keiner der Aktoren schreibend auf das Tupel zugreift. Sobald ein Aktor Schreibzugriff anfordert, wird eine Kopie des Tupel erstellt. Unsere Implementierung verwendet atomare Referenzzähler, die Race-Conditions ausschließt. Die dadurch entstehenden Laufzeitkosten sind minimal und dadurch zu vernachlässigen.

C. Scheduler

CAF hat zum Ziel, Millionen von Aktoren auf hunderten von Prozessorkernen effizient auszuführen. Das naive Verfahren jedem Aktor einen dedizierten Thread zuzuweisen würde zu hohem Aufwand bei geringer Skalierbarkeit führen, da jeder Thread seinen eigenen Stack allokiert und Ressourcen im Betriebssystemkern belegt. Um eine bessere Ausnutzung von Speicher und CPU-Zeit zu erreichen nutzt CAF eine beim Starten festgelegte Anzahl an Worker-Threads, über die dynamisch Aktoren verteilt werden. Die Anzahl der Worker wird zur Laufzeit bestimmt und richtet sich nach der lokalen Hardware-Nebenläufigkeit, wobei ein CPU-Thread einem Worker entspricht. Da das Scheduling der Aktoren oberhalb des Betriebssystems stattfindet, kann der Scheduler von CAF Aktoren nicht unterbrechen. Ein kooperativer Scheduler kann keine Fairness garantieren. Aktoren, die sich nicht zu einer kooperativen Ausführung eignen, beispielsweise weil sie auf blockierende I/O Funktionen zurückgreifen, können durch den Programmierer in separate Threads ausgelagert werden.

Um die Skalierbarkeit auch auf Mehrkern-CPUs sicherzustellen ist es wichtig, die Worker möglichst gleichmäßig auszulasten. Der Work-Stealing-Ansatz [12] ist ein grundlegendes Verfahren für Scheduling oberhalb des Betriebssystems für Fälle, in denen die Laufzeit keine Vorhersagen über das Verhalten der Anwendung treffen kann. Work-Stealing ist beispielsweise in Cilk [13], Intel Thread Building Blocks [14], sowie in Javas Fork/Join [15] implementiert. Jeder Worker hat seine eigene Job-Queue, die er abarbeitet solange Einträge in der Queue existieren. Sobald ein Worker keine Einträge mehr in seiner Job-Queue hat, nimmt er ("stiehlt") Einträge aus Job-Queues anderer Worker. Die Auswahl, bei welchem Worker gestohlen wird, geschieht in der Regel per Zufall.

III. Leistungsmessung in nebenläufigen Systemen

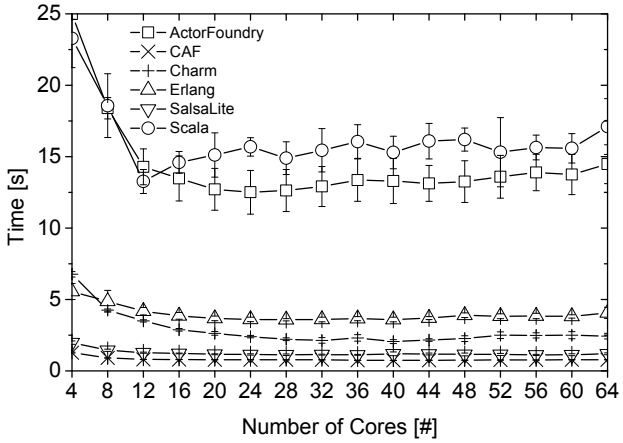
In diesem Abschnitt präsentieren und analysieren wir Messungen zu Laufzeit, Speicherverbrauch und Skalierbarkeit von CAF auf Mehrkernsystemen. Speicherverbrauch und Laufzeit wurden im Vergleich zu ActorFoundry, Erlang, SalsaLite, Scala (Akka) und Charm++ gemessen. Folgende Versionen wurden im Detail verwendet: (1) C++ mit CAF 0.13 (CAF) sowie Charm++ 6.5.1 (Charm), (2) Java mit dem auf Kilim [16] basierenden ActorFroundy 1.0 (ActorFoundry), (3) Erlang 5.10.2 mit HiPE für native Code Erzeugung und Optimierungslevel O3 (Erlang), (4) SALSA Lite 0.0.3 und (5) Scala 2.10.3 mit Akka. CAF und Charm++ wurden als Release-Version mit Clang 3.5.2 kompiliert. Scala, SALSA Lite und ActorFoundry liefen auf einer JVM mit maximal 10 GB RAM. Für die Kompilierung von ActorFoundry wurde der Java-Compiler in Version 1.6.0-38 verwendet, da aktuellere Compiler nicht unterstützt werden. Alle Graphen zeigen die Laufzeit als Mittelwert aus zehn unabhängigen Läufen. Der Speicherverbrauch wird als Boxplot dargestellt, um den stark fluktuierenden Speicherverbrauch über die Gesamtzeit der Ausführung zusammenzufassen. Das Hostsystem besitzt vier Opteron CPUs mit je 16 Kernen (insgesamt 64 CPU-Kerne). Um die Last stets gleichmäßig über alle CPUs zu verteilten wurde pro Durchlauf die Anzahl aktivierter CPU-Kerne um vier erhöht, beginnend bei 4. Der Programmcode zu allen Benchmarks ist online verfügbar unter `https://github.com/actor-framework`.

A. Erzeugung von Aktoren

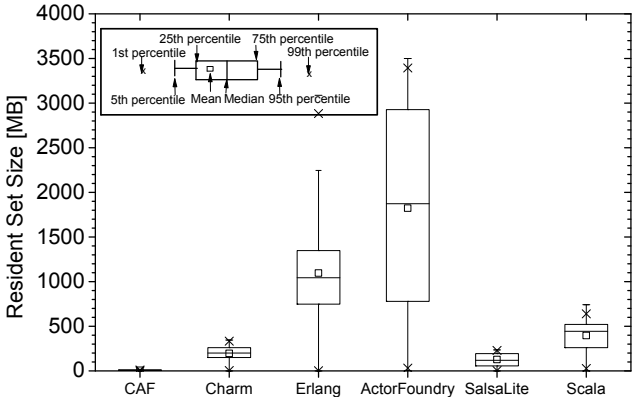
In unserem ersten Benchmark soll erfasst werden, wie effizient das Erzeugen vieler Aktoren in einem klassischen Divide-and-Conquer-Arbeitsablauf ist. Primär misst dieser Versuchsaufbau folglich die Kosten zum Starten eines Aktors. Dazu wurden rekursiv 2^{20} (etwa eine Million) Aktoren erstellt, die eine Baumstruktur bilden.

Die Laufzeitmessungen in Abbildung 2(a) zeigen, dass CAF die niedrigste Laufzeit benötigt. Nachdem ein globales Minimum erreicht wurde, bleibt die Laufzeit konstant. Salsa skaliert in diesem Testfall ähnlich wie CAF, hat jedoch eine höhere Laufzeit. Bei Charm nimmt die Laufzeit von vier bis 32 CPU-Kernen ab, stagniert dann, um bei 36 CPU-Kernen einen leichten Anstieg zu verzeichnen. Danach verschlechtert sich die Laufzeit von Charm. Erlang weist ähnliches Verhalten auf. Bei der Erhöhung der Anzahl der CPU-Kerne von vier auf 24 verbessert sich die Laufzeit, um dann zu stagnieren bzw. leicht anzusteigen. Scala skaliert von vier auf zwölf CPU-Kerne sehr gut, zeigt dann aber überraschenderweise eine signifikante Verschlechterung der Laufzeit. ActorFoundry erreicht sein Optimum bei 24 CPU-Kernen und hat bis dahin kontinuierlich seine Laufzeit verbessert. Ab 28 CPU-Kernen verschlechtert sich die Laufzeit von ActorFoundry jedoch wieder. Das erwartete Verhalten für alle Implementierungen in diesem Test ist ein konstanter Abfall in der Laufzeit, bis ein globales Minimum erreicht wird. Dieses Verhalten ist jedoch nicht bei allen Implementierungen zu beobachten. In absoluten Werten schneiden CAF und Salsa am besten ab.

Die Speichermessungen in Abbildung 2(b) zeigen, dass CAF mit ca. 10 MB den bei weitem niedrigsten Speicherverbrauch aufweist, gefolgt von Salsa und Charm mit jeweils ca. 250 MB, Scala mit 500 MB, Erlang mit gut einem GB Speicherverbrauch und ActorFoundry mit knapp zwei GB. Im Zusammenhang mit den vorherigen Laufzeitmessungen

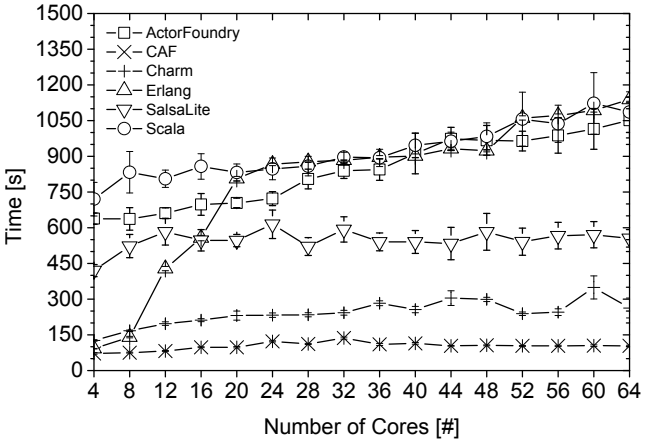


(a) Laufzeit

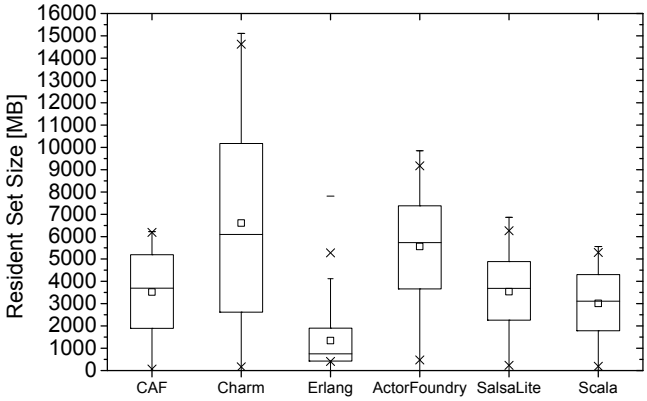


(b) Speicherverbrauch

Abbildung 2. Aktor spawn performance für 2^{20} Aktoren

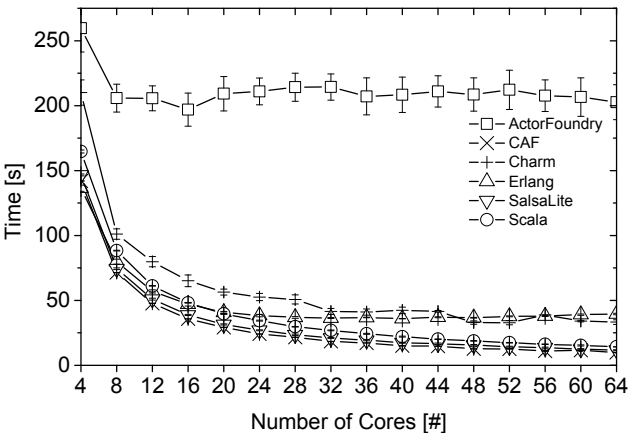


(a) Laufzeit

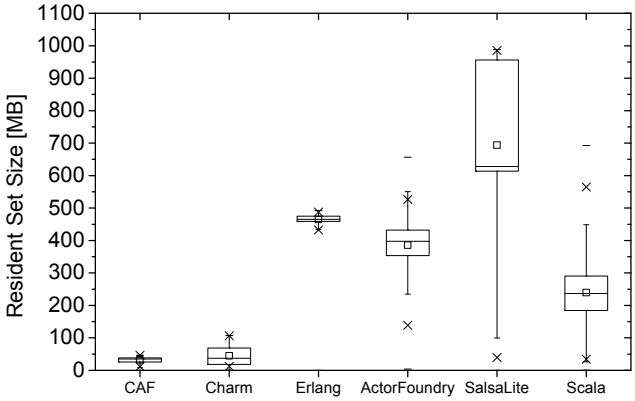


(b) Speicherverbrauch

Abbildung 3. Mailbox performance im N:1 Kommunikations Szenario



(a) Laufzeit



(b) Speicherverbrauch

Abbildung 4. Performance in einem gemischten Szenario mit zusätzlichem Work-Load

lässt sich vermuten, dass bei den langsameren Implementierungen ineffizientere Speicherverwaltung ausschlaggebend sein könnte. Dies würde auch die langsamere Laufzeit mit steigender Nebenläufigkeit erklären.

B. Mailbox unter Last

Mit unserem zweiten Testaufbau möchten wir das Skalierungsverhalten der Mailbox untersuchen. Dazu betrachten wir ein Szenario, bei dem ein zentraler Aktor Nachrichten von mehreren Sendern parallel empfängt, wodurch die Mailbox zum Synchronisationspunkt wird. Hierfür senden 100 Aktoren jeweils eine Million Nachrichten an einen einzigen Empfänger.

Das erwartete Verhalten für alle Implementierungen in diesem Test ist eine konstante oder leicht steigende Laufzeit, da mehr Kerne nicht das abarbeiten der 10^8 Nachrichten beschleunigen, sondern zu mehr Synchronisationsaufwand in der Mailbox führen. Abbildung 3(a) zeigt, dass CAF die niedrigste Laufzeit hat, welche beim Zuschalten mehrerer CPU-Kerne nur marginal ansteigt. Charm ist für wenig Nebenläufigkeit noch etwa gleichauf mit CAF, weist jedoch eine konstante Steigung und starke Fluktuation auf. Auch Erlang bietet auf vier CPU-Kernen eine gute Laufzeit, die in etwa der von CAF entspricht. Jedoch steigt die Laufzeit von Erlang ab zwölf CPU-Kernen unverhältnismäßig. Die Laufzeit von Salsa bleibt relativ konstant, benötigt jedoch im Minimum die dreieinhalbfache Laufzeit der schlechtesten Messung für CAF. ActorFoundry sowie Scala zeigen einen stetigen Anstieg der Laufzeit beim Hinzufügen von CPU-Kernen. Auch wenn Erlang für 4 und 8 Kerne noch zur Spitzengruppe gehört, so fällt doch auf, dass alle virtualisierten Lösungen signifikant langsamer als die beiden nativen Implementierungen sind. Die Messwerte des Speicherverbrauchs in Abbildung 3(b) zeigen, dass Erlang den geringsten Speicherverbrauch hat und maximal 8 GB an Arbeitsspeicher verwendet. CAF liegt im Schnitt bei 3,5 GB Speicherverbrauch, wobei maximal 6,5 GB verbraucht werden. Es folgt Scala mit ca. 3,5 GB Speicherverbrauch und maximal 6 GB, sowie Salsa mit 4 GB Speicherverbrauch und maximal 7 GB. ActorFoundry verbraucht im Mittel 6 GB und maximal 10 GB. Charm schlägt im Mittel mit 7 GB zu Buche, hat jedoch einen Maximalverbrauch von ca. 15,5 GB.

C. Gemischte Anwendung

In diesem Benchmark soll eine realistische Auslastung simuliert werden. Er besteht sowohl aus dem Austausch von Nachrichten sowie einer Primfaktor-Zerlegung. Aktoren wurden dabei in Ringen organisiert. Jeden Ring durchläuft eine Nachricht mit einem Zähler, der pro Durchlauf dekrementiert wird. Sobald der Zähler null erreicht, wird der Ring aufgelöst. Pro Ring gibt es einen Worker, der eine Primfaktor-Zerlegung (28.350.160.440.309.881 d.h. 329.545.133 und 86.028.157) vornimmt. Diese Zerlegung wird gestartet sobald ein Ring instanziiert wurde. Im Detail wurden 100 Ringe mit jeweils 100 Aktoren gestartet, der initiale Wert der Nachricht ist 1000, wobei jeder Ring insgesamt vier Mal neu instanziiert wurde. Das ideale Laufzeitverhalten für diesen Testaufbau ist ein konstantes Sinken der Laufzeit, bis ein globales Minimum erreicht ist.

Abbildung 4(a) zeigt, dass alle Implementierungen mit Ausnahme von ActorFoundry zum Ideal tendieren. Scala, Charm sowie ActorFoundry benötigen die längste Laufzeit. Erlang erreicht seine minimale Laufzeit auf 28 Kernen und bleibt danach stabil. Charm zeigt von 52 auf 56 CPU-Kernen einen leichten Anstieg, fällt dann bei 56 auf 60 aber wieder auf sein altes Niveau zurück. CAF hat in diesem Test erneut die kürzeste Laufzeit.

Die Messwerte zum Speicherbrauch in Abbildung 4(b) zeigen, dass CAF mit ca. 50 MB den geringsten Speicherverbrauch und die geringste Fluktuation aufweist. Es folgt Charm mit ca. 60 MB Speicherverbrauch und einem Maximum von knapp 130 MB. Scala folgt mit gut 250 MB und einem Maximum von 700 MB. ActorFoundry benötigt im Durchschnitt 400 MB Speicher und 700 MB als Maximum. Erlang zeigt kaum Fluktuation und verbraucht ca. 500 MB Speicher. Salsa bildet das Schlusslicht mit 700 MB sowie einem GB Maximalverbrauch.

IV. Leistungsmessung in verteilten und heterogenen Systemen

Nach den Messungen in nebenläufigen Szenarien möchten wir im Folgenden die Skalierbarkeit des Nachrichtenaustausches im Netzwerk sowie das Verhalten der Laufzeitumgebung beim parallelen Einsatz einer GPU messen.

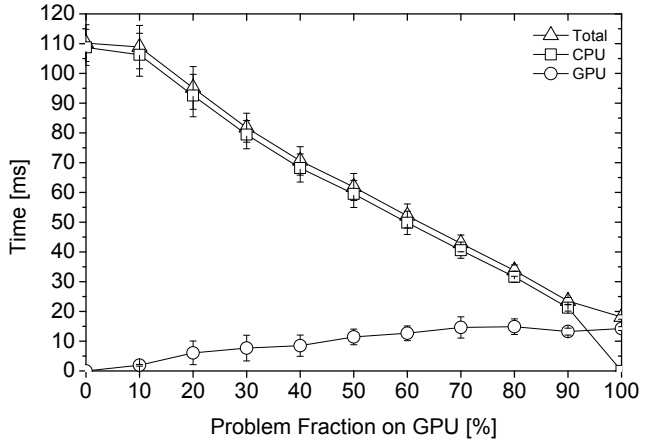
A. Kommunikations-Overhead: CAF vs. MPI

Für diesen Testaufbau haben wir virtuelle Maschinen (VMs) auf einem System mit 64 Kernen mit jeweils 2299 MHz ausgeführt. Zur Virtualisierung kam QEMU [17] zum Einsatz. Jede VM verfügt über einen CPU-Kern und 2GB Arbeitsspeicher. Die Kommunikation zwischen den VMs findet in einem lokalen VLAN statt, welches per Open vSwitch [18] eingerichtet ist. Zu jedem Testlauf gehören 4-64 Worker-VMs (in Viererschritten) sowie eine Master-VM, welche die Aufgaben auf die vorhandenen Worker verteilt. In diesem Test wurde nur die reine Rechenzeit gemessen, da sich die Initialisierungsphase zwischen CAF und OpenMPI stark unterscheidet.

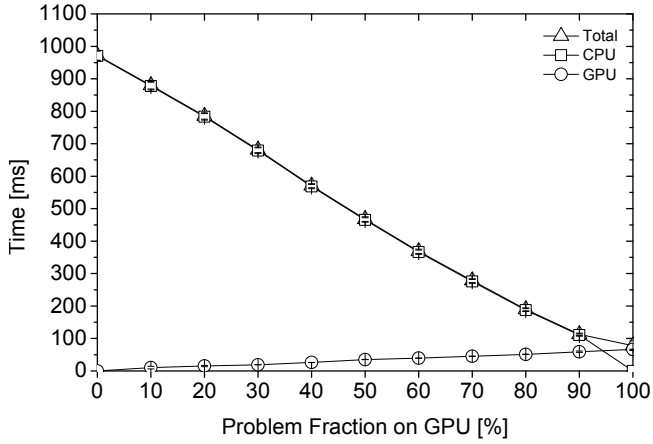
Abbildung 5 zeigt die Laufzeit in Sekunden als Funktion der verwendeten Worker-VMs. Zu beobachten ist ein stetiger Abfall der Laufzeit. Zu erwarten wäre eine Halbierung der Laufzeit bei gleichzeitiger Verdoppelung der Worker. Dieses Optimum wird von CAF und OpenMPI in den ersten beiden Skalierungsstufen (4 auf 8 sowie 16 auf 32) erreicht. Die Messungen zeigen, dass CAF trotz seiner höheren Abstraktion sogar ein leicht besseres Skalierungsverhalten als OpenMPI aufweist.

B. Hybride Berechnungen mit OpenCL-basierten Aktoren

In unserer letzten Leistungsmessung möchten wir zeigen, wie effizient sich eine GPU mit CAF in eine Aktor-basierte Anwendung integrieren lässt. Wir haben dazu die Mandelbrot-Menge als ein leicht zu parallelisierendes Problem gewählt und dieses über eine 12-Kern CPU und eine GPU (Nvidia



(a) Großes Problem: Mandelbrot mit max. 1200 Iterationen



(b) Kleines Problem: Mandelbrot mit max. 120 Iterationen

Abbildung 6. Workload von CPU auf GPU

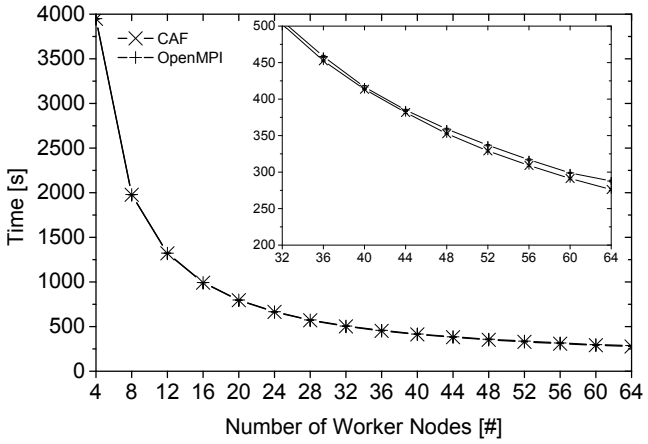


Abbildung 5. CAF vs MPI Mandelbrot

Tesla C2075) aufgeteilt, wobei wir in jedem Schritt 10% des Problems von der CPU auf die GPU verlagern. Das zu generierende Bild hat eine Auflösung von 1920x1080 Pixel und liegt im Bereich [-0.5 - 0.7375i - 0.1375i] des Mandelbrots. Beide Graphen in Abbildung 6 beschreiben die Laufzeit als Funktion über den Anteil an Berechnungen auf der GPU. Zusätzlich ist ein Fehlerbalken eingezeichnet, der die Standardabweichung anzeigt. Die Laufzeitwerte liegen in Millisekunden vor. Die Problemgröße in Abbildung 6(a) ist zehnmal größer als in Abbildung 6(b). Zusätzlich zur Gesamtlaufzeit werden die Zeiten für CPU und GPU (die Zeit zwischen Start und Terminierung) dargestellt, wobei die Gesamtlaufzeit nicht die Summe der CPU- und GPU-Laufzeit ist, da diese nebenläufig sind.

Allgemein ist die CPU-Laufzeit leicht niedriger als die Gesamtlaufzeit, wenn die CPU für Berechnungen verwendet wird. Im Vergleich benötigt die GPU nur einen Bruchteil der Laufzeit der CPU. Für beide Problemgrößen ist ein linearer Abfall der Laufzeit zu beobachten, wenn Teile auf die GPU ausgelagert werden. Die Fehlerbalken sind im Vergleich zur Laufzeit klein. In Abbildung 6(b) zeigt sich, dass durch die insgesamt kleinere Problemgröße die Auslagerungskosten an die GPU mehr ins Gewicht fallen. Zwar verringert sich auch hier die Laufzeit, jedoch weniger signifikant als in Abbildung 6(a) bei der größeren Problemgröße zu sehen ist. Ein Grund hierfür ist, dass sich CPU und GPU keinen Speicher teilen und der Großteil der Laufzeit auf das Kopieren der Daten entfällt.

V. Zusammenfassung und Ausblick

Die zunehmende Parallelisierung und Heterogenisierung der Hardware sowie der verstärkte Einsatz von Cloud-Technologien führen dazu, dass Entwickler für diese Einsatzgebiete hochskalierbare, robuste Lösungen benötigen. Das Aktorenmodell bietet ein integratives Konzept an, diese Teilaspekte im Verbund anzugehen, und es Anwendungen zu ermöglichen, nahtlos von Mehrkern- zu Cluster-Systemen überzugehen. Insbesondere im C++-Umfeld fehlt bisher jedoch ein vollwertiges Programmier-Framework, das die theoretischen Versprechen der Aktoren effizient in die Realität umsetzt.

Das hier vorgestellte C++ Actor Framework – CAF – basiert auf dem Aktorenmodell und richtet sich insbesondere an Infrastruktur-Software mit hohen Anforderungen an Skalierbarkeit und Robustheit. Darüber hinaus ist das Framework auch für Klein- und Kleinstgeräte, wie sie im IoT anzutreffen sind, geeignet. In den hier vorgestellten Leistungsmessungen zeigt sich, dass CAF nebenläufige Hardware sehr gut ausnutzt und insbesondere der Speicherverbrauch im direkten Vergleich zu konkurrierenden Ansätzen sehr gering ist.

Derzeit arbeiten wir an einer Portierung von CAF auf das IoT-Betriebssystem RIOT [19] und einer weiteren Reduktion des Speicherbedarfs. Darüber hinaus arbeiten wir an einer Ausweitung des Schedulings auf verteilte Systeme, um in massiv parallelen Installationen automatische Lastverteilung durch intelligente Migration von Aktoren zur Laufzeit zu gewährleisten.

LITERATUR

[1] J. E. Stone, D. Gohara, and G. Shi, "OpenCL: A Parallel Programming Standard for Heterogeneous Computing Systems," *IEEE Des. Test*, vol. 12, no. 3, pp. 66–73, May 2010.

[2] C. Hewitt, P. Bishop, and R. Steiger, "A Universal Modular ACTOR Formalism for Artificial Intelligence," in *Proceedings of the 3rd IJCAI*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1973, pp. 235–245.

[3] G. M. Amdahl, "Validity of the Single Processor Approach to Achieving Large Scale Computing Capabilities," in *Proceedings of the April 18-20, 1967, Spring Joint Computer Conference*, ser. AFIPS '67 (Spring). New York, NY, USA: ACM, 1967, pp. 483–485.

[4] D. Charousset, R. Hiesgen, and T. C. Schmidt, "CAF - The C++ Actor Framework for Scalable and Resource-efficient Applications," in *Proc. of the 5th ACM SIGPLAN Conf. on Systems, Programming, and Applications (SPLASH '14), Workshop AGERE!* New York, NY, USA: ACM, Oct. 2014.

[5] L. B. Kish, "End of Moore's law: Thermal (Noise) Death of Integration in Micro and Nano Electronics," *Physics Letters A*, vol. 305, no. 3–4, pp. 144–149, 2002.

[6] S. Meyers and A. Alexandrescu, "C++ and the Perils of Double-Checked Locking," *Dr. Dobb's Journal*, July 2004.

[7] J. Torrellas, H. S. Lam, and J. L. Hennessy, "False Sharing and Spatial Locality in Multiprocessor Caches," *IEEE Trans. Comput.*, vol. 43, no. 6, pp. 651–663, Jun. 1994.

[8] R. K. Karmani, A. Shali, and G. Agha, "Actor Frameworks for the JVM Platform: A Comparative Analysis," in *PPPJ*, 2009, pp. 11–20.

[9] L. V. Kale and S. Krishnan, "Charm++: Parallel programming with message-driven objects," *Parallel Programming using C++*, pp. 175–213, 1996.

[10] I.B.M. Corporation, "IBM System/370 Extended Architecture, Principles of Operation," IBM, Tech. Rep. SA22-7085, 1983.

[11] D. Charousset, T. C. Schmidt, R. Hiesgen, and M. Wählisch, "Native Actors – A Scalable Software Platform for Distributed, Heterogeneous Environments," in *Proc. of the 4rd ACM SIGPLAN Conference on Systems, Programming, and Applications (SPLASH '13), Workshop AGERE!* New York, NY, USA: ACM, Oct. 2013.

[12] R. D. Blumofe and C. E. Leiserson, "Scheduling Multithreaded Computations by Work Stealing," *J. ACM*, vol. 46, no. 5, pp. 720–748, Sep. 1999.

[13] R. D. Blumofe, C. F. Joerg, B. C. Kuszmaul, C. E. Leiserson, K. H. Randall, and Y. Zhou, "Cilk: An Efficient Multithreaded Runtime System," *SIGPLAN Not.*, vol. 30, no. 8, pp. 207–216, Aug. 1995.

[14] Pheatt, Chuck, "Intel® threading building blocks," *Journal of Computing Sciences in Colleges*, vol. 23, no. 4, pp. 298–298, 2008.

[15] Lea, Doug, "A Java fork/join framework," in *Proceedings of the ACM 2000 conference on Java Grande*. ACM, 2000, pp. 36–43.

[16] S. Srinivasan and A. Mycroft, "Kilim: Isolation-Typed Actors for Java," in *Proceedings of the 22nd ECOOP*, ser. LNCS, vol. 5142. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 104–128.

[17] F. Bellard, "QEMU, a Fast and Portable Dynamic Translator," in *USENIX Annual Technical Conference, FREENIX Track*, 2005, pp. 41–46.

[18] Pfaff, Ben and Pettit, Justin and Amidon, Keith and Casado, Martin and Koponen, Teemu and Shenker, Scott, "Extending Networking into the Virtualization Layer." in *Hotnets*, 2009.

[19] E. Baccelli, O. Hahm, M. Günes, M. Wählisch, and T. C. Schmidt, "RIOT OS: Towards an OS for the Internet of Things," in *Proc. of the 32nd IEEE INFOCOM. Poster*. Piscataway, NJ, USA: IEEE Press, 2013.

Performability analysis of an unreliable M/M/1-type queue

Ruslan Krenzler, Hans Daduna
Department of Mathematics, University of Hamburg
20146 Hamburg, Germany

***Abstract*—We consider a single server exponential queue with infinite waiting room and state dependent service rates. The server is unreliable and wears down when active and additionally may break down randomly. Repair times are random as well. The system is controlled with the possibility of preventive maintenance after a fixed number of services. This number is subject of optimization under a prescribed general cost structure. In case of breakdown (i.e. under repair) or during preventive maintenance no service is possible and no new customers are admitted.**

We determine explicitly the stationary distribution of the queueing-availability process and show that the system is separable, i.e. the steady state is of product form.

MSC 2000 Subject Classification: Primary 60K; Secondary: 60J10, 60F05, 60K20, 90B22, 90B05

Keywords: Queueing systems, random breakdown and repair, product form steady state, availability, reactive and preventive maintenance, loss systems.

I. Introduction

We report in this paper about our research on bridging the gap between the areas of performance analysis and reliability, resp. availability analysis. We carry out this inside of the world of product form models, i.e. we will come up with a **separable model**, i.e. the stationary distribution is of product form.

Product form networks of queues are common models for easy to perform structural and quantitative first order analysis of complex networks in Operations Research applications. The most prominent representatives of this class of models are the Jackson [1] and Gordon-Newell [2] networks and their generalizations as BCMP [3] and Kelly [4] networks, for a short review see [5].

We consider in this paper the simplest building block of these networks, the M/M/1 queue and couple it with reliability considerations, i.e. our topic in this paper is performability analysis, the interplay of performance and dependability. To be more precise, we analyse an abstract degradable service system which wears down during each service. As a consequence the failure probability increases. After the system breaks down it is repaired and thereafter resumes work as good as new. To prevent from breakdowns, the system will be maintained after a fixed maximal number of services since the most recent repair or maintenance. During repair or maintenance the system is blocked, i.e., no service is provided and no new job may join the system. Rejected customers are *lost* to the system. We will provide explicit steady state probabilities for this system which encompass both, the availability characteristics and the queue behavior, and is of **product form**, i.e. shows decomposition properties. Using this result we discuss optimization of preemptive maintenance policies.

The mathematical description of this model is a time homogeneous Markovian two-dimensional vector process, where the first coordinate represents the behaviour of the queue, while the second coordinate describes the availability-degradation status of the server.

Product form networks of queues are characterized by the fact that in steady state (at any fixed time t) the joint distribution of the multi-dimensional (over nodes) queueing process is the product of the marginal distributions of the isolated (non Markovian) queues in steady state. With respect to the research described in this note the key point is that the coordinates of this vector process represent objects of the same class, namely queueing systems. On the other side our two-dimensional vector process has two coordinates which describe fundamentally different processes.

Our main result (Theorem 1) is that in these systems where service and availability strongly interact, asymptotically and in equilibrium (at fixed time instants) the queueing process and the availability process seem to decouple, which means that a product form equilibrium emerges, similarly to the Jackson network processes' equilibrium. We show that this simple formula opens the way to efficiently perform cost analysis and a subsequent optimization of the system's parameter setting.

Related research

In [6], networks of queues with unreliable servers were investigated which admit product form steady states in twofold way: the joint queue length vector of the system (which in general is not a Markov process) is of classical product form as in Jackson's Theorem and the availability status of the nodes as a set valued supplementary variable process constitutes an additional product factor attached to the joint queue length vector.

We show that for the case of a single server with a more complex unreliability (break down) behavior than in [6] a product form steady state distribution can be obtained as well.

Because we can consider the degradation of the server (at least partly) as originating from external influences (the environmental conditions) the investigation of this report can be subsumed under the heading of queues in a random environment. In fact, it can therefore be considered as a special case

of the class of models investigated in [7]. In [7] and its more detailed predecessor [8] we considered a single server system with infinite waiting room in a random environment. The service system and the environment interact in both directions: the environment process may block the service process, and the server, on the other side, may control the environment process by changing its state at service completion epochs. Our main contribution here is to show that for the specific performability context of the unreliable queue the theoretical framework of [8] is not needed. Furthermore, we show some typical insights concerning performability issues in this area, which have been surprising to us.

The unreliable queuing system investigated in this paper can also be viewed as a quasi-birth-death (QBD) process where the level describes the number of customers in the queue and the phase describes the availability-degradation status. When the service rate of the unreliable system is constant, it is a level-independent (or homogeneous) QBD process, otherwise it is a level-dependent (or inhomogeneous) QBD process. The classical methods to analyse QBD process are the matrix-geometric methods, popularized by Neuts. See e.g. [9].

The system in this paper is a QBD process, where the level and phase distribution in the steady state are independent for fixed time instants. In [10, Chapteer 15], Latouche and Ramaswami analysed a special case of such processes: level-independent with finite number of phases. In [11], Ramaswami and Taylor analysed more general – provided some irreducibility conditions are satisfied – level-dependent and independent QBD processes with countable number of phases.

The modeling approach we follow in this paper can be termed "state space model", as classified in [12, Section 2.2], where a general survey is given on the area of performability analysis. In this survey especially pointers to the relevant literature are provided. We mention here only the book [13], which provides fundamental introductions into main subfields of performability.

II. The general model

We consider a two-dimensional process $(X, Y) = ((X(t), Y(t)) : t \in [0, \infty))$ with state space $E = \mathbb{N}_0 \times K$. The M/M/1-queueing process $X = (X(t) : t \in [0, \infty))$ has state space $\mathbb{N}_0$, whereas the degradation status, resp. the availability of the server is recorded by $Y = (Y(t) : t \in [0, \infty))$ with state space K. The server in the system is a degradable single server under First-Come-First-Served regime (FCFS) with infinite waiting room. The arrival stream of customers is Poisson with rate $\lambda > 0$.

The availability-degradation space of the process is partitioned into disjoint components $K := K_W + K_B$, where[1] $K_W = \{0, 1, ...N-1\}$ with $N \in \mathbb{N}$ indicates the degradation status of the server if it is up. The degradation status is determined by the number of completed services since the most recent repair, resp. maintenance. We consider the situation where performance of the server is degrading when serving customers.

$Y(t) = 0$ indicates that the server at time t is as good as new, i.e. just after a repair or preventive maintenance. N is the admissible (maximal) number of services before maintenance is mandatory again. So, if the degradation status is $Y(t) = N-1$, after finishing the next service maintenance is necessary.

The *non-availability states* are $K_B = \{b_m, b_r\}$, where b_m indicates an ongoing preventive maintenance, while b_r indicates an ongoing repair after breakdown of the server. The general rule is that whenever the availability-degradation process enters K_B the service process is **b**locked, which is resumed only whenever the process Y returns to $0 \in K_W$, and the server **w**orks again. Whenever the server is down, i.e. the availability-degradation process stays in K_B, new arrivals are lost.

The system develops over time as follows.

1) If the server is as good as new at time t, then the availability-degradation process indicates this by taking value $Y(t) = 0 \in K_W$ and if there are $X(t) = n$ customers in the queue then service is provided to the customer at the head of the queue with rate $\mu(n) > 0$. When his service expires he departs and the availability-degradation process increases to $Y(t) = 1 \in K_W$.

2) If the server at time t has provided service to $k < N$ customers since the most recent repair or maintenance – this is indicated by $Y(t) = k \in K_W$ – and after finishing the ongoing service at time $t + s$ $X(t)$ decreases to $X(t + s) = X(t) - 1$, while the performance status increases to $Y(t + s) = k + 1$ if $k + 1 < N$. If $k + 1 = N$, the server undergoes immediately a complete maintenance, which is indicated by $Y(t + s) = b_m \in K_B$. Time for maintenance is exponentially distributed with mean ν_m^{-1} and after maintenance the server is as good as new.

3) Additionally, the server is under risk to break down if it is not under maintenance or repair. If the server at time t has provided service to $Y(t) = k < N$ customers since the most recent repair or maintenance it may break down with performance status dependent rate $\nu_k \geq 0$. It is assumed that an idling server ($X(t) = 0$) is held in warm standby and (if $Y(t) = k$) breaks down with rate $\nu_k \geq 0$ as well. Recall that degradable means that the ν_k are increasing in k.

4) If the availability-degradation process at time t is in state $Y(t) = b \in K_B$ no service is provided to customers in the queue and arriving customers are lost.

From this description the non negative transition rates of (X, Y) are for $(n, k) \in E$

$$\begin{aligned}
q((n,k),(n+1,k)) &= \lambda, && k \in K_W,\\
q((n,k),(n-1,k+1)) &= \mu(n) && k \leq N-2, n \geq 1,\\
q((n,N-1),(n-1,b_m)) &= \mu(n), && n \geq 1,\\
q((n,k),(n,b_r)) &= \nu_k \in \mathbb{R}_0^+, && k \in K_W, n \geq 1,\\
q((n,b_m),(n,0)) &= \nu_m \in \mathbb{R}_0^+, && n \geq 0,\\
q((n,b_r),(n,0)) &= \nu_r \in \mathbb{R}_0^+, && n \geq 0,\\
q((n,k),(i,m)) &= 0\,, && \text{otherwise for}\\
& && (n,k) \neq (i,m)\,.
\end{aligned}$$

[1]For notational convenience we assume throughout $N \geq 2$. The case $N = 1$ is simpler, but incorporating it requires notational burden.

The diagonal elements of the transition intensity matrix $Q := (q((n,k),(i,m)) : (n,k),(i,m) \in E)$ are determined by the requirement that the row sums are 0.

ASSUMPTION. (X,Y) is throughout a homogeneous strong Markov jump process with cadlag paths which is irreducible and non-explosive in finite times.

III. STEADY STATE DISTRIBUTION

Our aim is to compute the steady state distribution of (X,Y). The main result is

Theorem 1. *The queueing-availability process (X,Y) is ergodic if and only if*
$C := \sum_{n=0}^{\infty} \prod_{i=1}^{n} \frac{\lambda}{\mu(i)} < \infty$ *holds.*

If (X,Y) is ergodic its unique stationary and limiting distribution π is given as
$\pi(n,k) = \xi_\mu(n)\theta_N(k)$, *for $(n,k) \in E$ with normalizing constant C, $\mu := (\mu(i) : i = 1,2,\dots)$ and*

$$\xi_\mu(n) := C^{-1} \prod_{i=1}^{n} \frac{\lambda}{\mu(i)}$$

$$\theta_N(k) := \prod_{i=1}^{k} \left(\frac{\lambda}{\nu_i + \lambda}\right)^i \theta_N(0), \qquad 0 \le k \le N-1$$

$$\theta_N(b_m) := \frac{\lambda}{\nu_m}\theta(N-1) = \frac{\lambda}{\nu_m} \prod_{i=1}^{N-1} \left(\frac{\lambda}{\nu_i + \lambda}\right)^i \theta_N(0)$$

$$\theta_N(b_r) := \left(\frac{(\nu_0 + \lambda)}{\nu_r} - \frac{\lambda}{\nu_r} \prod_{i=1}^{N-1} \left(\frac{\lambda}{\nu_i + \lambda}\right)^i\right) \theta_N(0)$$

$$\begin{aligned}\theta_N(0) := &\left(\sum_{k=0}^{N-1} \prod_{i=1}^{k} \left(\frac{\lambda}{\nu_i + \lambda}\right)^i \right.\\ &+ \frac{\lambda}{\nu_m} \prod_{i=1}^{N-1} \left(\frac{\lambda}{\nu_i + \lambda}\right)^i \\ &\left. + \left(\frac{(\nu_0 + \lambda)}{\nu_r} - \frac{\lambda}{\nu_r} \prod_{i=1}^{N-1} \left(\frac{\lambda}{\nu_i + \lambda}\right)^i\right)\right)^{-1}\end{aligned}$$

Before proving the theorem some remarks seem to be in order.

The steady state distribution of an ergodic birth-death process with birth rate λ and death rates $\mu(n)$ is

$$\xi = \left(\xi(n) := C^{-1} \prod_{i=1}^{n} \frac{\lambda}{\mu(i)} : n \in \mathbb{N}_0\right). \tag{III.1}$$

But our marginal process $X = (X(t) : t \ge 0)$ (having this steady state as well) is in general not a birth-death process - in general it is not even Markov. Clearly, Y is not Markov for its own as well.

This observation is remarkable not only because X is not a birth-death process, but also because under queue length i neither λ is the effective arrival rate (expected number of arrivals per time unit) nor are the $\mu(i+1)$ the effective service rates (expected maximal number of departures per time unit) for queue length i. The conclusion is that both rates are diminished by the influence of the non-availability of the server by the same portion. The precise formula for effective arrival rate in equilibrium is given in Proposition 3.

The advantage and the surprising consequence of the special form of ξ in (III.1) is that we can use the same formulas for average number of customers and empty-queue probability as for a plain queueing system with Poisson input rate λ, with exponential service rates $\mu(i)$, and without environment. This includes well known formulas in literature for $M/M/1$, $M/M/c$ and $M/M/\infty$ queues.

Insensitivity property of the system

As long as ergodicity is maintained, the availability-degradation indices of the system can change drastically without changing the steady state of the queue length at any fixed time point. An intuitive interpretation of this result seems to be hard. Especially, this insensitivity can not be a consequence of the form of the availability-degradation process.

The insensitivity does not imply, that the time development of the queue length processes with fixed λ and $\mu(n)$ is the same under different breakdown and repair behavior. This can be seen by considering the stationary sojourn time distribution of admitted customers, which strongly depends on the interruption time distributions (= sojourn time distribution of Y in K_B). Similarly, multidimensional stationary probabilities for $(X(t_1), X(t_2), \dots, X(t_n))$ depend on the occurrence frequency of the event $(Y \in K_B)$.

Although the theorem is a special case of Theorem 1 in [7], we provide for completeness a sketch of the proof which due to the special case under consideration is more transparent.

Proof. of Theorem 1 The global balance equations of the Markov process (X,Y) are for $(n,k) \in E$ with $1 \le k \le N-1$

$$\begin{aligned} &\pi(n,k)\left(\lambda + \nu_k + 1_{[n\ge1]}\mu(n)\right) \\ = \;& \pi(n-1,k)1_{[n\ge1]}\lambda \\ &+\pi(n+1,k-1)\mu(n+1), \end{aligned} \tag{III.2}$$

$$\begin{aligned} &\pi(n,0)\left(\lambda + \nu_0 + 1_{[n\ge1]}\mu(n)\right) \\ = \;& \pi(n-1,0)1_{[n\ge1]}\lambda \\ &+\pi(n,b_r)\nu_r + \pi(n,b_m)\nu_m \end{aligned} \tag{III.3}$$

$$\pi(n,b_r)\nu_r = \sum_{k=0}^{N-1} \pi(n,k)\nu_k \tag{III.4}$$

$$\pi(n,b_m)\nu_m = \pi(n+1,N-1)\mu(n+1) \tag{III.5}$$

Inserting the proposed product form solution for $(n,k) \in E$

$$\pi(n,k) = \xi_\mu(n)\theta_N(k) = C^{-1} \prod_{i=1}^{n} \frac{\lambda}{\mu(i)} \theta_N(k),$$

yields after appropriate cancelling a **finite** set of equations for $(\theta_N(k) : k \in K)$

$$\begin{aligned} \theta_N(k)\left(\lambda + \nu_k + \cancel{1_{[n\ge1]}\mu(n)}\right) &= \cancel{\theta_N(k)1_{[n\ge1]}\mu(n)} \\ &\quad +\theta_N(k-1)\lambda, \\ \theta_N(0)\left(\lambda + \nu_0 + \cancel{1_{[n\ge1]}\mu(n)}\right) &= \cancel{\theta_N(0)1_{[n\ge1]}\mu(n)} \\ &\quad +\theta_N(b_r)\nu_r + \theta_N(b_m)\nu_m \\ \theta_N(b_r)\nu_r &= \textstyle\sum_{k=0}^{N-1} \theta_N(k)\nu_k \\ \theta_N(b_m)\nu_m &= \theta_N(N-1)\lambda, \end{aligned} \tag{III.6}$$

which can be solved directly. The product form of this solution leads to the conclusion that the values of the $\theta_N(k)$ do not matter for deciding whether $\prod_{i=1}^{n} \frac{\lambda}{\mu(i)} \theta_N(k)$ is summable. This leads to the given ergodicity criterion. □

Remark 2. The special case of Theorem 1, where the service rate μ is constant, can be proved using [10, Theorem 15.1.3] with $\eta := \lambda/\mu$.

Another insensitivity property

The last finite set of equations (III.6) defines a generator equation for some intensity matrix $\widetilde{Q}$. The proof of Theorem 1 reveals that the solution of that equation $\theta \cdot \widetilde{Q} = 0$ (see (III.6)) does not depend on the values $\mu(n)$. So, changing the service capacity of the queueing system will not change the steady state of the availability behaviour, as long as the system remains stable (ergodic).

Limiting models

There are two extremal cases of the model: $N = \infty$ and $\mu = \infty$. These cases can be considered as limiting models.

(i) If we set $N = \infty$ there is no preventive maintenance and we arrive at a simple M/M/1/∞ with unreliable server where the breakdown rates depend on the time since the most recent repair. A precise model is obtained from the following non-negative transition rates

$$\begin{aligned} q((n,k),(n+1,k)) &= \lambda, && k \in K_W, \\ q((n,k),(n-1,k+1)) &= \mu(n) && n \geq 1, \\ q((n,k),(n,b_r)) &= \nu_k \in \mathbb{R}_0^+, && k \in K_W, n \geq 1, \\ q((n,b_r),(n,0)) &= \nu_r \in \mathbb{R}_0^+, && n \geq 0, \\ q((n,k),(i,m)) &= 0\,, && \text{otherwise for} \\ & && (n,k) \neq (i,m)\,. \end{aligned}$$

on state space $\mathbb{N}_0 \times K$ with, $K = K_W + K_B$, $K_W := \mathbb{N}_0$, and $K_B = \{b_r\}$.

To apply the previously calculated results, we need an additional criterion for ergodicity

$$\sum_{k=0}^{\infty} \prod_{i=1}^{k} \left(\frac{\lambda}{\nu_i + \lambda} \right)^i < \infty. \tag{III.7}$$

(ii) If we set $\mu(n) = \infty$ for all n we get a simple reliability model with preventive replacement after a fixed number of N shocks which a system experiences. Here a single shock is generated by wear out through a single service. The queuing part of the system disappears and we get a stochastic process living on the space K, described by a generator $\widetilde{Q}$, the same as the previously mentioned one, and with steady state distribution $(\theta(k) : k \in K)$ exactly the same as θ in Theorem 1.

We can consider the equality of both θs as an extension of the insensitivity property. The other way round, one can use a system with instantaneous service, i.e. $\mu = \infty$, to calculate the marginal distribution θ for a system with finite $\mu(n)$. In [14], the authors use this approach for a certain queueing system in a random environment which is an inventory.

IV. Cost analysis

We are interested in the effects of the unreliability and non-availability on associated cost imposed on the service system. We consider the following costs (per time unit):

- c_r repair costs per unit time after a sudden breakdown,
- c_m maintenance costs per unit time for a preventive maintenance,
- c_b loss costs per customer ($\equiv$ breakdown costs) and unit time,
- c_w waiting costs per customer per unit time.

The cost function per time unit in state (n,k) is

$$f(n,k) = \begin{cases} c_m + c_b + c_w \cdot n, & k = b_m, n \geq 0 \\ c_r + c_b + c_w \cdot n, & k = b_r, n \geq 0 \\ c_w \cdot n, & k \in K \setminus \{b_r, b_m\}, n \geq 0. \end{cases}$$

The average costs for an ergodic system can be calculated as

$$\begin{aligned} \bar{f}_T(N,\mu) = & \frac{1}{T} \int_0^T f\left(X_\omega(t), Y_\omega(t)\right) \\ & \xrightarrow{T \to \infty} \sum_{(n,k)} f(n,k)\pi(n,k) =: \bar{f}(N,\mu), \qquad P.a.s \end{aligned}$$

Using the product form stationary distribution of the unreliable service system we get

$$\begin{aligned} \bar{f}(N,\mu) = & \left(c_m + c_b\right) \theta_N(b_m) + \left(c_r + c_b\right) \theta_N(b_r) \\ & + \underbrace{\sum_{n=1}^{\infty} (n c_W) \xi_\mu(n)}_{\text{independent of } N}. \end{aligned}$$

$\bar{f}(N,\mu)$ can be interpreted as steady-state expected reward rate [12, Section 2.2] or as the instantaneous reward rate of the system in steady state, as it is defined e.g. in [15, Section 2, p. 200].

This expression makes optimization, whenever possible, of the preventive maintenance by control of N a rather easy task although π is still complicated. Because the last sum of $\bar{f}(N,\mu)$ is independent of N and the first two terms are independent of μ, we may reduce our investigation to the function

$$g(N) = \left(c_m + c_b\right) \theta_N(b_m) + \left(c_r + c_b\right) \theta_N(b_r)$$

defined on a set of instances $M \subset \mathbb{N}$.

Remark. If g does not have a global minimum, the optimal solution of the problem is not to maintain at all. This may happen for example if $M := \mathbb{N}_0$ and g is strictly decreasing.

V. Applications

We will shortly describe different applications where our integrated queueing-availability-degradation model can contribute to performability assessment.

Averaging failure rates

Due to lack of data about state dependent failure rates an intuitive procedure is to use in the evaluation process for planning a system as first approximation fixed constant failure rates $\bar{\nu}_f$. Clearly, the best such "first order approximation" would be to use averaged failure rates.

$$\bar{\nu}_f = \sum_{k=0}^{N-1} \theta(k)\nu_k \tag{V.1}$$

Unfortunately the formula (V.1) - requires knowledge of the probabilities $\theta(k)$ $k \in \{0, \ldots, N-1\}$. Nevertheless, let us assume, we have this exact average at hand.

To study the influence of averaging the failure rate we consider the following example: The parameters are $\lambda = 1$, $c_m = 1$, $c_r = 2$, $c_b = 1$, $\nu_m = 0.3$, $\nu_r = 0.1$. We restrict the number of successive services to $M := \{1, \ldots, 40\}$ to keep the number of optimization instances large enough but finite. That means, at least after 40 services we will preventively maintain the system.

Although this example is fictional, we consider it as plausible, because the cost structure is typical. Planned (proactive) maintenance is cheaper than reactive repair: $c_m = 1 < 2 = c_r$, and its duration is shorter: mean times $1/\nu_m < 1/\nu_r$.

For linear failure rates $\nu_k = 0.01 \cdot k, k = 0, \ldots, N-1$ we see from Figure V.1a that the optimal maintenance policy is set with $N = 6$.

Averaging the failure rates yields for all $k \in K_W$: $\nu_k = \bar{\nu}_f = 0.04368593$. The resulting cost function is shown in Figure V.1b. Consequently, the optimal maintenance policy is to perform preventive maintenance after $40 = \max(M)$ services. Even more, if we remove the restriction $\max(M) = 40$, the optimal solution of the system is: not to maintain the system.

Remark. The remarkable difference between the system with state dependent failure rates and its version with averaged fixed failure rates is not an exception. In [8, Corollary 3.16] we proved that for ergodic systems with constant positive failure rates only three possible behavior of cost function $g(\cdot)$ exist

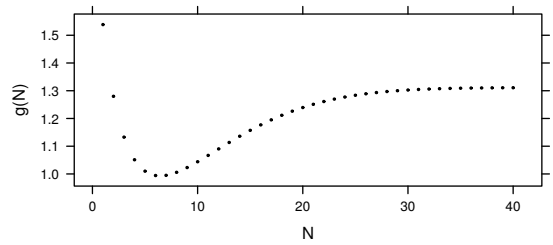
(i) strictly monotone increasing
(ii) constant
(iii) strictly monotone decreasing

It follows: In case (i) the optimal policy is to maintain as soon as possible. In case (ii) we may maintain the system after any number of services. In case (iii) the optimal policy is to maintain as late as possible if M - the set of allowed N's - is bounded and never to maintain if this set is not bounded.

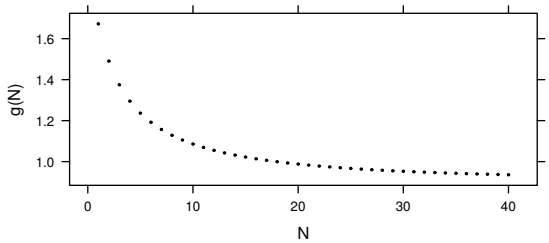
CONCLUSION. The examples show that to extend results and conclusions obtained from systems with fixed constant failure rate (even under optimal approximation) to predict behaviour of more general systems with varying failure rates may be risky.

Performability

From [16, p.8, Def.1.5] Performability: The probability that the system reaches an accomplishment level over a utilization interval $(0, t)$. That is, the probability that the system does a certain amount of useful work over a mission time t.



(a) Original system with linear failure rate $\nu_k = 0.01 \cdot k$



(b) Approximated system with averaged (constant) failure rate $\nu_k \equiv \bar{\nu_f} = 0.04368593$

Figure V.1: Cost functions g with $\lambda = 1$, $c_m = 1$, $c_b = 1$, $c_r = 2$, $\nu_m = 0.3$, $\nu_r = 0.1$.

For stationary systems we can and will take without loss of generality always $t = 1$. The surprising fact of our next finding is, that as long the system is ergodic, the relevant probability does depend on λ, but not on the service rates $\mu(n)$.

Proposition 3. *For an ergodic system as in Theorem 1 in equilibrium the probability that the server can treat an offered load of size λ per time unit is $1 - \theta(b_m) - \theta(b_r)$.*

Therefore, the effective throughput of the system is $\lambda \cdot (1 - \theta(b_m) - \theta(b_r))$. This value is also equal to the effective arrival rate.

Proof. For $T \to \infty$ by ergodicity

$$T^{-1} \int_{0,T} 1_{[Y(t) \in \{0,1,\ldots,N-1\}]} d(t) \to (1 - \theta(b_m) - \theta(b_r))$$

This implies that in stationary setting during one time unit the system is up with probability $(1-\theta(b_m)-\theta(b_r))$ and the mean production rate is $\lambda \cdot (1 - \theta(b_m) - \theta(b_r))$ □

Remark 4. Using similar formulas as for throughput in Proposition 3, we can calculate further system parameters in equilibrium:

- The average number of failures in unit of time, which is the same as the average number of time instants per unit of time when repair is finished, is

$$\theta_N(b_r)\nu_r. \tag{V.2}$$

- The average number of started maintenances in unit of time, which is the same as the average number of time instants per unit of time when a maintenance is finished, is

$$\theta_N(b_m)\nu_m. \tag{V.3}$$

As mentioned in the introduction, our investigations in the previous sections were on the single building block of the networks in [6] but we allow for a more complicated breakdown and repair process as that investigated in [6].

Remark 5. In this paper we assumed that the failure rates ν_k are increasing in k to mimic effects of wearing out, but we never used this property in our proofs. Thus all the previously calculated formulas and derived statements remain valid also for monotone decreasing or even general $(\nu_k : k \in \{1, \ldots, N-1\})$.

Software rejuvenation.

Software rejuvenation is a proactive and preventive solution to handle transient software failures, complementing the reactive recovering mechanisms applied to breakdowns of a software package. After an initial almost failure-free "longevity interval" an application (exiting out of this interval) experiences that software failures typically accumulate over time of usage. The problem is to estimate the time point after the last rejuvenation when proactive restarting from a clean internal state would reduce cost of recovering. An early study is [17] where Markovian models are used to assess the possible benefits of software rejuvenation under the assumption of exponential dwell times, e.g. for the duration of the longevity interval. Vaidyanathan and Trivedi [18] refined this model by incorporating the actual workloads the application is faced with into the aging process of the application software. They underpinned their proposal by a case study.

The proposed procedure is: Evaluate the relevant workload clusters, their occurrence and duration statistics, and the transition probabilities between clusters. Estimate the slope of the aging in every cluster, and finally compute the average slope in equilibrium of the (Markovian) cluster sequence. With this averaged slope determine the (mean) time to exhaustion of the resources, i.e. the time to failure.

In terms of reliability processes this is the replacement policy "replace at planned time", see e.g. [19, Section 6.4].

If we notice that degrading (aging) of the application (which we may consider as a virtual server) is connected to sequentially arriving jobs, which bring in the different workload requests, an alternative replacement policy would be "replace after a prescribed number of shocks", where a "shock" is induced by a served job.

Following the gross construction of the average slope in [18] it would be adequate to construct an average job: take the mean service times of workload clusters weighted by their occurrence frequency over all jobs as mean service time request μ^{-1} of a job. Estimate the occurrence rate λ of jobs.

The connection to the replacement policy "replace at planned time" of Vaidyanathan and Trivedi [18] is then computed as follows.

Take the model of Section II with $\lambda < \mu$ constant. Neglect as in [18] the immediate failures by setting $\nu_k = 0$ for all k which is justified by the assumption that we are in the longevity interval after restart.

Assume, the outcome of the estimated time to exhaustion of the resources is Δ, which is just the time between two rejuvenations. Then between two rejuvenations the application is busy for an amount of time $\Delta \cdot \rho$, where $\rho = \lambda/\mu$ is the utilization of the application. During this time the application serves $N := \frac{\Delta \cdot \rho}{\mu^{-1}} = \Delta \cdot \lambda$ jobs. This is exactly the N in the model of Section II and determines the policy "replace after a prescribed number of shocks". See [19].

References

[1] J. Jackson, "Networks of waiting lines," *Operations Research*, vol. 5, pp. 518–521, 1957.

[2] W. Gordon and G. Newell, "Closed queueing networks with exponential servers," *Operations Research*, vol. 15, pp. 254–265, 1967.

[3] F. Baskett, M. Chandy, R. Muntz, and F. Palacios, "Open, closed and mixed networks of queues with different classes of customers," *Journal of the Association for Computing Machinery*, vol. 22, pp. 248–260, 1975.

[4] F. Kelly, "Networks of queues," *Advances in Applied Probability*, vol. 8, pp. 416–432, 1976.

[5] H. Daduna, "Stochastic networks with product form equilibrium," in *Stochastic Processes: Theory and Methods*, ser. Handbook of Statistics, D. Shanbhag and C. Rao, Eds. Amsterdam: Elsevier Science, 2001, vol. 19, ch. 11, pp. 309–364.

[6] C. Sauer and H. Daduna, "Availability formulas and performance measures for separable degradable networks," *Economic Quality Control*, vol. 18, pp. 165–194, 2003.

[7] R. Krenzler and H. Daduna, "Loss systems in a random environment: steady state analysis," *Queueing Systems*, vol. 80, no. 1-2, pp. 127–153, 2015. [Online]. Available: http://dx.doi.org/10.1007/s11134-014-9426-6

[8] ——, "Loss systems in a random environment," http://arxiv.org/abs/1312.0539, Dec. 2013.

[9] M. Neuts, *Matrix Geometric Solutions in Stochastic Models - An Algorithmic Approach*. Baltimore, MD: John Hopkins University Press, 1981.

[10] G. Latouche and V. Ramaswami, Eds., *Introduction to Matrix Analytic Methods in Stochastic Modeling*, ser. ASA-SIAM Series on Statistics and Applied Probability. Philadelphia,: SIAM, 1999.

[11] V. Ramaswami and P. Taylor, "An operator analytic approach to product-form networks," *Communications in Statistics. Stochastic Models*, vol. 12, no. 1, pp. 121–142, 1996. [Online]. Available: http://dx.doi.org/10.1080/15326349608807376

[12] K. Trivedi, E. Andrade, and F. Machida, "Chapter 1 - combining performance and availability analysis in practice," in *Dependable and Secure Systems Engineering*, ser. Advances in Computers, A. Hurson and S. Sedigh, Eds. Elsevier, 2012, vol. 84, pp. 1 – 38.

[13] B. Haverkort, R. Marie, G. Rubino, and K. Trivedi, *Performability Modeling, Technique and Tools*. New York: Wiley, 2001.

[14] A. Krishnamoorthy and V. C. Narayanan, "Stochastic decomposition in production inventory with service time," *European Journal of Operational Research*, vol. 228, pp. 358–366, Jul. 2013.

[15] K. S. Trivedi, J. K. Muppala, S. P. Woolet, and B. R. Haverkort, "Composite performance and dependability analysis," *Performance Evaluation*, vol. 14, no. 3-4, pp. 197 – 215, 1992, performability Modelling of Computer and Communication Systems.

[16] G. Bolch, S. Greiner, H. de Meer, and K. S. Trivedi, *Queueing networks and Markov chains*, 2nd ed. Hoboken, New Jersey: John Wiley, 2006.

[17] Y. Huang, C. Kintala, N. Kolettis, and F. N. Dudley, "Software rejuvenation: Analysis, module and applications," in *Proceedings of the Twenty-Fifth International Symposium on Fault-Tolerant Computing*, ser. FTCS '95. Washington, DC: IEEE Computer Society, 1995, pp. 381–390.

[18] K. Vaidyanathan and K. S. Trivedi, "Workload-based estimation of resource exhaustion in software systems," in *Performability Modeling of Computer and Communication Systems*, ser. Arbeitsberichte des Instituts für Informatik, R. German, J. Lüthi, and M. Telek, Eds., vol. 34. Universität Erlangen-Nürnberg, 2001, pp. 126–130.

[19] T. Nakagawa, *Stochastic Processes with Applications to Reliability Theory*, ser. Springer Series in Reliability Engineering. London: Springer, 2011.

Liste der Beitragenden

Georg Carle, TU München *27, 37*
Dominik Charousset, HAW Hamburg *83*
Hans Daduna, Universität Hamburg *1, 90*
Paul Emmerich, TU München *27*
Raphael Hiesgen, HAW Hamburg *83*
Mario Hock, Karlsruher Institut für Technologie *46*
Helge Jürgensen, Nextragen GmbH *75*
Ruslan Krenzler, Universität Hamburg *90*
Paul J. Kühn, Universität Stuttgart *18*
Ming Li, TU Hamburg-Harburg *55*
Sadaf Momeni, Universität Hamburg *63*
Daniel Raumer, TU München *27, 37*
Peter Reichl, Universität Wien *23*
Torsten M. Runge, Universität Hamburg *37*
Thomas C. Schmidt, HAW Hamburg *83*
Dominik Scholz, TU München *27*
Andreas Timm-Giel, TU Hamburg-Harburg *55*
Phuong Nga Tran, TU Hamburg-Harburg *55*
Marian Triebe, HAW Hamburg *83*
Tadeus Uhl, FH Flensburg *75*
Sonja Weichler, Universität Hamburg *63*
Mario Willems, Karlsruher Institut für Technologie *46*
Florian Wohlfart, TU München *27, 37*
Bernd E. Wolfinger, Universität Hamburg *37, 63*