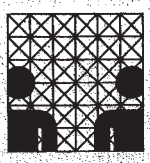
Universität Hamburg
Department Informatik
Vogt-Kölln-Str. 30
D-22527 Hamburg

Bericht 281

Leistungs-, Zuverlässigkeits- und Verlässlichkeitsbewertung von Kommunikationsnetzen und verteilten Systemen

4. GI/ITG-Workshop
MMBnet 2007
13./14. September 2007

herausgegeben von
Bernd Wolfinger,
Klaus Heidtmann



Vorwort

Zum vierten Mal, nach 1998, 2002 und 2005 – ergo mit stetig kürzer werdenden Abständen – findet nunmehr in Hamburg der Workshop *"Leistungs-, Zuverlässigkeits- und Verlässlichkeitsbewertung von Kommunikationsnetzen und verteilten Systemen"* (MMBnet) statt, der im Namen der GI/ITG-Interessengruppe "Messung, Modellierung und Bewertung von Rechensystemen (MMB)" wiederum am Department Informatik der Universität Hamburg durch die Arbeitsgruppe Telekommunikation und Rechnernetze (TKRN) veranstaltet wird. Der vorliegende Tagungsband enthält die schriftlichen Ausarbeitungen zu den Vorträgen, die im Rahmen des MMBnet2007-Workshops am 13./14. September 2007 präsentiert werden.

Kommunikationsnetze und verteilte Systeme gewinnen zunehmend an Bedeutung. Besonders deutlich wird diese Tendenz bei Hochgeschwindigkeitsnetzen, dem Internet, der Mobil- und Echtzeitkommunikation. Selbst die enorme Leistungssteigerung bei der Rechnerhardware und vor allem bei der Datenübertragung machen Bewertungen solcher Systeme nicht überflüssig, sondern erfordern in verstärktem Maße neue Mess-/Modellierungsmethoden und -werkzeuge, u.a. zur Berücksichtigung der Dienstgüte (QoS), des Performance-Managements und steigender Anforderungen an die Zuverlässigkeit und Verfügbarkeit.

Ziel des Workshops sollte es deshalb sein, Weiterentwicklungen der methodischen Grundlagen und Werkzeuge zur Messung und Modellierung aufzuzeigen und über Erfahrungen bei interessanten Anwendungsstudien (zu Rechnernetzen und/oder verteilten Anwendungen) zu berichten. Gewünscht waren im Programmaufruf somit auch innovative Beiträge, die eine Brücke schlagen zwischen den bislang noch häufig getrennten "Welten" der Leistungs- und der Zuverlässigkeitsbewertung. Der Workshop richtet sich primär an Nachwuchswissenschaftler (insbesondere Doktoranden) mit F&E-Aktivitäten im MMB-Umfeld.

Die für den MMBnet2007-Workshop angestrebten Themenschwerpunkte betrafen bei den

- *methodischen Grundlagen und Werkzeugen:*
  - formale Beschreibungstechniken (zur Spezifikation von System- und Lastmodellen),
  - Last-/Verkehrsmodellierung, -charakterisierung, “Traffic Engineering“,
  - analytische, simulative, hybride Modellauswertung,
  - Modellierungswerkzeuge, Netzemulatoren, Lastgeneratoren,
  - Messmethoden und -werkzeuge,

sowie bei den

- *Anwendungsbereichen (bezogen auf die Bewertung):*
  - Internet, Intranets,
  - Mobilkommunikation/Drahtlose Netze (WLANs, Sensornetze, UMTS) und "Ubiquitous Computing",
  - Echtzeitkommunikation (insbes. Voice-over-IP, ’Triple Play’, QoS-Bewertungen),
  - ‘Peer-to-Peer’-Anwendungen,
  - multimediale Anwendungen, (z.B. in den Bereichen E-Learning, Telemedizin, Verkehrstelematik, Videokonferenzen und Verteiltes CSCW, Video-Streaming), etc.

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

- Netzwerkkalkül („Network Calculus“)
- Lastklassifikation, -modellierung und -transformation
- Analytische Modellierung
- Simulative Modellierung, Messungen und Verkehrsanalysen
- Dienstgüte (QoS) und ihre Bewertung.

Darüber hinaus wurde ein eingeladener Vortrag (Dr. J. Charzinski, München) in das Workshop-Programm aufgenommen.

Als Motto des Workshops haben die Veranstalter, wie bereits bei den Vorgänger-Workshops, wiederum den Slogan *"Vom Internet zum Wartenetz"* gewählt wegen seiner inhärenten Zweideutigkeit: Einerseits hat sich das Internet infolge der enormen Zuwachsraten bei der Anzahl von Benutzern und angeschlossenen Endsystemen – zumindest in manchen Teilbereichen/Subnetzen – trotz der während der vergangenen Dekade signifikanten Kapazitätsvergrößerungen in den "Backbone"-Netzen, aus Endbenutzersicht nicht selten zu einem "Wartenetz" entwickelt (im Sinne großer Wartezeiten für Benutzer bei Zugriffen auf Web-Server, Informations- und Abfragedienste, bei umfangreichen Dateitransfers oder auch der teilweise noch immer relativ stark limitierten Dienstgüte/QoS bei Echtzeitkommunikationsdiensten wie Voice-over-IP und IP-TV, etc). Andererseits ist es ein Anliegen der MMB-"Community", geeignete Modelle, z.B. "Wartenetze" (nunmehr im mathematischen Sinne), zu finden und – durch deren intelligente Anwendung und Nutzung – die Wartesituationen in derart komplexen Gebilden wie dem Internet oder öffentlicher Mobilnetze zu entschärfen bzw. langfristig möglichst weitgehend zu vermeiden. Wir wünschen uns, dass der MMBnet2007-Workshop auch in diesem Sinne einen nützlichen und praxisrelevanten Beitrag leistet.

Unser besonderer Dank gilt den Mitgliedern des Programmkomitees und den zusätzlichen Gutachtern für die zügige und sorgfältige Begutachtung der eingereichten Papiere sowie den Vortragenden für die termingerechte Erstellung der Endversionen ihrer Beiträge. Auch für die Bereitschaft von Herrn Dr. Joachim Charzinski (Nokia Siemens Networks), ohne zu zögern das Angebot der Tagungsveranstalter anzunehmen, einen eingeladenen Vortrag zu halten, möchten wir uns an dieser Stelle nochmals ganz herzlich bedanken. Die tatkräftige Unterstützung der Workshop-Vorbereitung und -Durchführung seitens Frau Katrin Köster und Frau Margit Wichmann verdient überdies lobende Erwähnung. Nicht zuletzt möchten wir auch unserem eigenen Department Dank sagen für die Unterstützung des Workshops insbesondere durch die Publikation des Tagungsbandes als Department-Bericht.

Wir hoffen, dass die bewusste Beschränkung des Teilnehmerkreises auf ausschli*eßlich aktive Beitragende* zum MMBnet-Workshop – d.h. auf die (Ko-)Autoren der akzeptierten Beiträge sowie die beteiligten TPC-Mitglieder – zu sehr intensiven Diskussionen während des Workshops führen wird. Überdies möge der erwünschte Know-how-Transfer unter den TeilnehmerInnen mit hoffentlich zahlreichen wissenschaftlichen Anregungen für (praxis-) relevante und innovative Forschungsaktivitäten gelingen (z.B. für laufende Promotionsvorhaben im Sinne des durch uns angestrebten Doktorandenforums). Und, 'last but not least', wünschen wir, dass die Weltstadt Hamburg durch ihr reichhaltiges kulturelles, Erholungs- und Vergnügungsangebot den Teilnehmern auch über den Workshop hinaus einen sehr angenehmen Aufenthalt im hohen Norden der Republik ermöglichen wird.

Hamburg, im September 2007 Prof. Dr. Bernd E. Wolfinger, PD Dr. Klaus D. Heidtmann

Mitglieder des Programmkomitees:

- **Bolch, Gunter** (Univ. Erlangen)
- **Carle, Georg** (Univ. Tübingen)
- **Daduna, Hans** (Univ. Hamburg)
- **De Meer, Hermann** (Univ. Passau)
- **Drobnik, Oswald** (Univ. Frankfurt)
- **Echtle, Klaus** (Univ. Duisburg-Essen)
- **German, Reinhard** (Univ. Erlangen)
- **Heidtmann, Klaus-D.** (Univ. Hamburg)
- **Killat, Ulrich** (TU Hamburg-Harburg)
- **Krieger, Udo R.** (Univ. Bamberg)
- **Kühn, Paul J.** (Univ. Stuttgart)
- **Lehmann, Axel** (Univ. der Bundeswehr München)
- **Müller-Clostermann, Bruno** (Univ. Duisburg-Essen)
- **Siegle, Markus** (Univ. der Bundeswehr München)
- **Spaniol, Otto** (RWTH Aachen)
- **Tavangarian, Djamshid** (Univ. Rostock)
- **Uhl, Tadeus** (FH Flensburg)
- **Walke, Bernhard** (RWTH Aachen)
- **Wolfinger, Bernd E.** (Univ. Hamburg) *(Vorsitz)*

Weitere Gutachter:

- **Gollmann, Dieter** (TU Hamburg-Harburg)
- **Heckmüller, Stephan** (Univ. Hamburg)
- **Wüchner, Patrick** (Univ. Passau)

MMBnet 2007-Workshop

„Leistungs-, Zuverlässigkeits- und Verlässlichkeitsbewertung von Kommunikationsnetzen und Verteilten Systemen“

13./14. September 2007
Universität Hamburg, Department Informatik

Programm (1. Tag)

Donnerstag, den 13. September 2007

13:00 – 13:15 **Begrüßung und Vorstellung der Teilnehmer**

13:15 – 14:15 **Eingeladener Beitrag**
J. Charzinski (Nokia Siemens Networks, München):
Admission Control versus Dimensioning for Modern Network Services

14:20 – 15:30 **„Network Calculus / Netzwerkkalkül“**

- K. Angrishi, U. Killat (TU Hamburg-Harburg):
  Analysis of a Real-Time Network using Statistical Network Calculus with Effective Bandwidth and Effective Capacity

- U. Klehmet, T. Herpel, K.-S. Hielscher, R. German (Universität Erlangen-Nürnberg):
  Worst Case Analysis for Multiple Priorities in Bitwise Arbitration

15:30 – 16:15 Pause

16:15 – 18:00 **„Lastklassifikation, -modellierung und -transformation“**

- S. Heckmüller, B. E. Wolfinger (Universität Hamburg):
  Modellierung verlustinduzierender Lasttransformationen für markovsche Ankunftsprozesse

- B. Müller-Clostermann, M. Tilev (Universität Duisburg-Essen):
  Macroscopic Workload Management and Mainframe Capacity Planning

- D. Versick, H. Kopp, D. Tavangarian (Universität Rostock):
  Ein Lastmodell für die Definition von I/0-Lasten beim Zugriff auf Sekundär-speicher paralleler Systeme

19:30 Gemütliches Beisammensein

Programm (2. Tag)

Freitag, den 14. September 2007

9:15 – 10:30 **„Analytische Modellierung“**

- K. Tippner (Universität Hamburg):
  Rates of convergence for closed cycles of Infinite Server Queues

- F. Brenner, B. Müller-Clostermann (Universität Duisburg-Essen):
  Synchronised Concurrent Processes as a Modelling Pattern and Compositional Uniformisation: Work in Progress

10:30 – 11:00 Pause

11:00 – 12:45 **„Simulative Modellierung, Messungen und Verkehrsanalysen“**

- M. Schinnenburg, R. Pabst, K. Klagges, B. Walke (RWTH Aachen):
  A Software Architecture for Modular Implementation of Adaptive Protocol Stacks

- I. Dietrich, C. Sommer, F. Dressler, R. German (Universität Erlangen-Nürnberg):
  Automated Simulation of Communication Protocols Modeled in UML 2 with Syntony

- G. Münz, S. Li, G. Carle (Universität Tübingen):
  Traffic Anomaly Detection Using K-Means Clustering

12:45– 14:00 Mittagspause

14:00 – 15:45 **„Dienstgüte (QoS) und ihre Bewertung“**

- A. Kolesnikov (Universität Hamburg):
  Konzeption und Entwicklung eines echtzeitfähigen Lastgenerators für Multimedia-Verkehrsströme in IP-basierten Rechnernetzen

- T. Uhl, L. Diederichsen (FHS Flensburg & ITD Informationstechnologie, Holdorf):
  Quality of Service (QoS) bei IPTV

- M. Kiritz (Universität Hamburg):
  Untersuchung der Fehlerrobustheit H.264/AVC-kodierter Videoströme bei simulierten Übertragungsverlusten

15:45 – 16:00 **Abschlussdiskussion / ”Workshop Summary”**

Inhalt

Dienstgüte (QoS) und ihre Bewertung

Admission Control versus Dimensioning for Modern Network Services

Joachim Charzinski, Nokia Siemens Networks – Joachim.Charzinski@nsn.com

Abstract

Internet applications are evolving into networked services with increasing traffic demand. At the same time, real-time applications are being moved from dedicated circuit switched networks onto IP based networks. We discuss a number of issues that arise when quality of service is to be delivered for those new applications over IP.

1 Introduction

The Internet has been growing steadily for several decades, not giving any guarantees of resources to applications but offering a design that makes connectivity resilient to link outages and that is open to simply introducing new applications without making changes to the network. The main changes that were regularly made to the networks composing the Internet were periodic capacity upgrades to support growing traffic demands.

Lately, in the course of unifying network technologies towards converged "next generation" networks, a number of applications such as voice telephony, TV distribution and video on demand services have been moved from dedicated networks providing data rate guarantees onto IP (Internet Protocol) based infrastructures.

This article discusses some of the latest developments in applications, their impact on resource management in IP networks and ways to handle resource management for those services.

Sec. 2 discusses the evolution of network capacity and applications towards "Web 2.0" and interactive real-time applications. Network regions and their specific problems are highlighted in Sec. 3 before network dimensioning approaches are presented in Sec. 4. Finally, a number of mechanisms for admission control and results comparing capacity requirements with and without admission control are presented in Sec. 5.

2 Evolution of Network Capacity and Applications

2.1 Growth

The amount of traffic carried in the world's networks has been growing significantly for a number of years. A number of components stimulate this growth.

Number of users
The number of people using the Internet is growing. There is still a vast number of people on the earth who do not use the Internet. On the other hand, Internet services in principle offer great opportunities also in developing countries. Market analysts expect that by 2015, $5{\cdot}10^9$ people will be using the Internet, many of them not through a multi-purpose computer but rather through simple-to-use mobile devices. Ubiquitous availability of mobile network access is a further driver – users can spend more time connected to the Internet as they are no longer restricted to the location of their access lines.

Traffic Demand per Application and New Applications
Along with other growth factors, the availability of high data rates and new ideas for application design stimulate the growth of traffic demands per application. In addition, new applications are coming up that are using the Internet. The Web (together with search engines) caused significant growth in the late 1990s, e-commerce Web sites and file sharing sustained the growth and today's "Web 2.0" mash-ups are driving the growth further. Application trends are described in more detail in Sec. 2.2.

Data Rates on Access and Core Links
Data rates on residential access lines (as well as wireless access) have been steadily increasing for several decades. As both the number of access lines in the world and the data rate per access line are increasing, the increase of overall traffic is faster than both components, which is reflected by a corresponding evolution of core network router interface speeds and data rates on core links.

Number of Servers
An ever increasing number of servers for the same services (such as Web or e-commerce sites), as well as new services (such as the currently famous YouTube, flickr or Second Life) are increasing the choice of communication partners that users have. Correspondingly, also the number of search, recommendation and indexing services increases, which in turn are another cause for traffic increase both on the servers and in the networks.

Extension of Usage Scenarios
For the past ten years, in addition to reaching more and more users and serving them with increasing access data rates, the Internet has found its way from a work-related communications facility into a general-purpose network that users employ for entertainment and social contact and communication as well as business purposes.

2.2 Evolution of Applications

Structure, programming and communications models of web based services and applications have dramatically changed since the invention of the Web in 1989. In the beginning, a Web page was designed to represent a document and hyperlinks were used only to navigate from a document to another document the first one referred to. Later developments saw documents being composed of images and text parts coming from different servers, advertisements dynamically delivered to match a page's topic or a user's search terms.

Latest developments – partially summarized under the term "Web 2.0" include services ("mash-ups") being composed of a number of contributing services that are all accessed by a web browser using the HTTP protocol. Dynamic behavior is included on client and server side to improve reaction times on user input.

Fig. 1 shows the servers involved in serving the findr [1] and Helsinki bus map [2] mash-ups. Findr is a combination of tag associations and flickr (a photography sharing service). The Helsinki bus map combines map graphics from google maps with location information showing where the public transport buses are at a certain moment.

Both mash-ups get the primary page and application structure from a primary server (shaded in Fig. 1) and retrieve additional information from further servers.

2.3 Evolution of Traffic Characteristics and Performance Measures

The Quality of Service (QoS) users perceive for classical Web applications, e-commerce or e-business applications is determined by the time to complete a transaction, i.e., the latency between user input and the completion of the request displayed on the Web browser. Browsers have evolved into loading multiple elements on a page through parallel HTTP/TCP

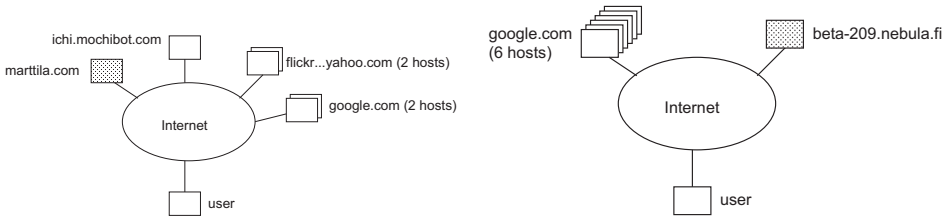


Figure 1: Client-server relations for the findr (left) and hklkarttan mash-ups (right).

connections to improve performance, even if the elements are coming from the same server. The element size and connection volume distributions reported previously [3] have significantly changed with applications' development towards Web 2.0. First measurements suggest that there are more and more small elements downloaded from servers, e.g. for pre-loading information or partial page refresh functions. These elements are often carried within the same HTTP/TCP connection, such that the duration of connections increases. On the other hand, the size of the really big elements downloaded on the Web has increased (e.g., the size of software patches or media files).

As refreshing information on a Web page does no longer lead to complete page re-loads, Web designers employ partial page refresh mechanisms more often, leading to continuous data streams being fed into Web browsers without users interacting with the browser. As an example, the Helsinki bus map mash-up described above produced a sustained data rate of 5.1 kbit/s upstream (for regular GET requests) and 3.6 kbit/s downstream (for the responses) for the bus location updates of two buses over five minutes of passively watching the scene.

Whereas for the "elastic" downloading or transaction applications a certain mean data rate is required, users experience only gradual reduction of QoS if the available data rate is lower. In the past years, a number of services has been moved onto IP based infrastructures (partially the Internet, partially separate IP networks) that require guarantees of a certain minimum data rate delivered. Under the label of "convergence" and "next generation networks", interactive real-time services such as telephony, video conferencing and TV distribution are entering IP based networks. Those services traditionally were offered by networks that explicitly supported guaranteed data rates through circuit switching techniques. On IP based infrastructures they require at least careful dimensioning support in order to deliver the expected Quality of Service (QoS).

In addition, the control systems for real-time applications are also response time critical – just consider acceptable call set-up latencies for voice communication. As communication, community and search applications will converge on the Internet, also the control functions will become part of mashed-up services.

3 Network Regions and their Specific Problems

Fig. 2 shows a systematic sketch of the different network regions involved in delivering traffic over the Internet. In normal communication across the Internet, a number of networks need to cooperate, starting with a user's home network, continuing through access and aggregation network, an Internet Service Provider's (ISP) network, possibly an Internet exchange node connecting several ISPs' networks, finally crossing the world through one or more of the "tier 1" core networks, which are interconnected through peering links, to finally arrive at the server site which again might have a network of its own.

On the current Internet, packets are just forwarded through such a chain of networks. The number of parties involved and the multitude of communication relations to be supported renders everything beyond simple connectivity a challenging task. Explicit reservations for QoS (Quality of Service) support are difficult to realize in some of the different

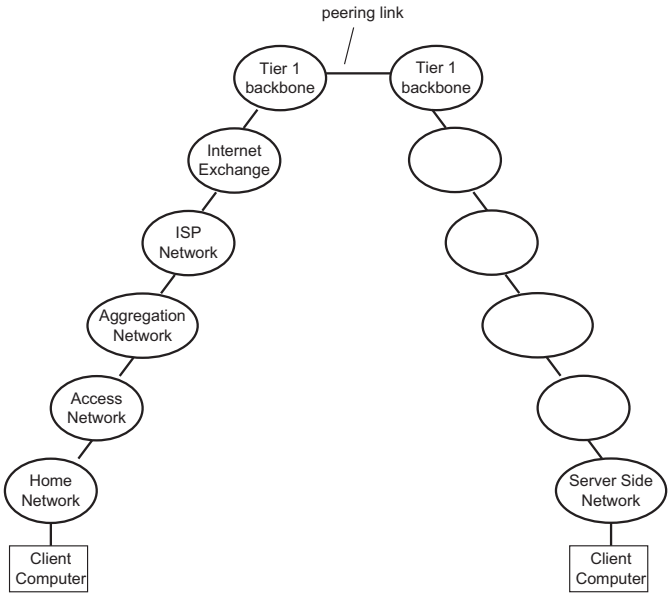


Figure 2: Typical multi-domain Internet architecture between client and server.

network regions for a number of technical reasons that will be outlined in more detail in the following sections.

An additional non-technical but very important inhibitor for introducing explicit reservations and QoS support across network domains is the fact that the Internet structure does not offer mechanisms for financial compensation for supporting priority treatment of packets in remote networks.

3.1 Home Network Issues

Most residential users are not interested or able to perform advanced configurations or checks on their home networks to ensure sufficient performance to support QoS for demanding applications. Therefore the most attractive solution for QoS support in home networks is to provide sufficient total capacity such that the most relevant traffic mixes will not suffer from any QoS degradation problems. To further complicate the situation, there is no single home networking technology that would fit all installation conditions. Whereas WLANs (wireless local area networks) are the easiest technology to deploy for residential users, their reach is too limited for some homes and in some other homes there is too much concurrent activity on the WLAN spectrum that limits achievable throughputs. Cable oriented technologies like Ethernet cabling, plastic optical fiber based solutions, phone or TV cable based solutions are often not possible because the corresponding outlets' locations do not coincide with the desired placement of equipment and long cables are unwanted in living rooms. Broadband powerline communication is a feasible alternative in many homes but finds its limits in environments with strong noise from electrical appliances.

3.2 Access and Aggregation Networks

In access and aggregation networks, network capacity is expensive and links are usually operated at relatively low economy of scale. As will be shown in Sec. 5.3, admission control and overprovisioning approaches are both inefficient for low bundle sizes. Especially for services like IPTV or video on demand, which have strong correlation in request arrivals and may exhibit holding times of several hours, an additional problem of admission

control is that once a request is been blocked, the probability of being accepted at a re-trial some seconds or even some minutes later is very low. Thus, there must be enough capacity provisioned such that blocking occurs extremely rarely.

3.3 Inter-Domain Issues

The inter-domain region is where the lack of payment structures for QoS differentiation is most obvious. Whereas it would be simple to offer multiple priority classes and mark packets accordingly even at network borders, there is no economic incentive for a neighbor network to provide high priority treatment for selected packets. In addition, in a long chain of networks operated by different providers, it is very hard for an end system to check if the contracted QoS treatment has actually been delivered. If an application gets insufficient data rate or experiences too much packet loss or delay, on the one hand it is non-trivial to identify the guilty network provider in the chain and on the other hand it is even more difficult to get compensation from a remote provider.

Despite all these problems, Internet video on demand services such as YouTube [4] are on-line and heavily used – however, not according to the classical pay-per-view business model, in which users would probably stop using the service once the first paid movie did not play with sufficient quality. An explanation for this can be found in the qualitative graph shown in Fig. 3.

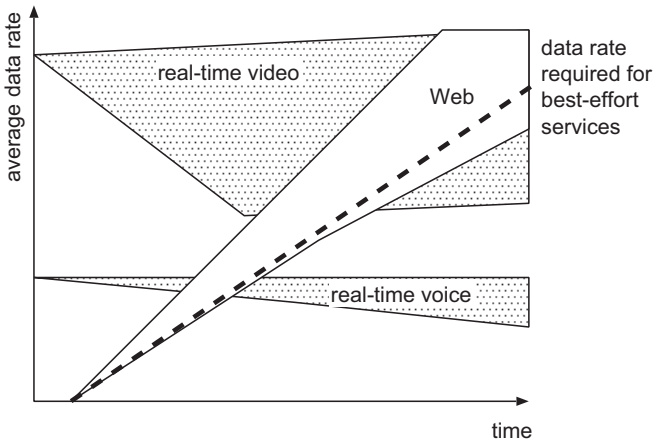


Figure 3: Evolution of data rates on the Internet.

The timeline in Fig. 3 shows the evolution of required data rates for voice and video services as well as for standard Web access over time (several years). Whereas the data rate requirements for basic voice or video services are relatively constant or even decreasing, access to today's web pages requires mean data rates that are a factor of 10 higher than 10 years ago. Consequently, there is a point in time when the data rate required to view a normal web page in acceptable time is higher than the data rate required to transport a Voice over IP (VoIP) conversation, and the VoIP service can therefore be carried on the Internet without additional QoS measures in most cases. In well-connected parts of the Internet, we are currently at the point where also efficiently encoded videos can be viewed in real time over the Internet without further precautions. Some of the current drivers behind the Web data rate evolution were discussed in Sec. 2.3.

There are scenarios that do not suffer from the inter-domain issues for resource reservations: Services like commercial Voice over IP, IPTV or video on demand are often provided by a combined network and service provider that operates the access network, the service and a regional network connecting access and service. In those cases, the above arguments of missing business cases do not hold and explicit resource reservation schemes like

admission control can actually be employed to accelerate the introduction of new capacity-demanding services.

4 Network Dimensioning

The network dimensioning and traffic management methods used in the classical PSTN (public switched telephone network) or mobile networks differ significantly from the methods used currently on the Internet.

As more and more services are moving from specialized infrastructures such as the PSTN onto IP based infrastructures, it is an interesting question what dimensioning and traffic management approach to use for those "converged" networks – independent if they are part of the Internet or separate IP based networks.

4.1 PSTN and Mobile Networks

The PSTN guarantees a given data rate for all accepted connections, mostly by associating a fixed data rate channel with a switched circuit. Admission control is performed link-by-link. As the traffic model for voice traffic is well-defined (fixed constant data rate for ongoing connections together with well-described holding time and connection inter-arrival time distributions), the corresponding systems and links can easily be dimensioned for a tolerable level of blocking probability, e.g., $p_B = 10^{-3}$. Connections between different network operators may be engineered for a higher level of blocking probabilities, but most of them also guarantee the data rate per connection.

In the case of a link outage, either an underlying physical layer recovery mechanism restores connectivity quickly, or connections carried over a lost link are dropped, but users can immediately set up a new connection again. In case of increased offered load, e.g. due to hot spot or high load scenarios, the blocking rate increases but still all admitted flows can carry their respective data rates.

4.2 Current Internet

The Internet consists of a large number of organizationally separate networks that are interconnected by Internet exchanges or peering links (see Sec. 3).

In the access area, static overbooking is often employed. Only a fraction of the sum of all customer peak rates is deployed as capacity in the access and aggregation networks. E.g., 1000 DSL (digital subscriber line) users with 1 Mbit/s lines would be served by a common 100 Mbit/s line when a factor of 10 is applied for overbooking. Different overbooking factors are applied by different access providers and for different traffic profiles.

In core networks, the capacity on wide area links is mostly dimensioned according to observed traffic demand. Long-term link loads are monitored and if certain thresholds are exceeded, the corresponding links are upgraded. This approach of dealing with limited capacity has also been characterized as "throw bandwidth at the problem", as it invests into additional network capacity rather than control technology to avoid capacity shortage. The fact that there is no bandwidth control in the core networks allows networks to simply reroute traffic in case of link outages without having to restore any reservation information. If rerouting happens fast, users might not even notice problems with their applications.

An additional measure often employed to increase efficiency of network operation is to optimize routing in core networks according to a measured traffic matrix. Changes in routing can be realized in the core network by employing overlay mechanisms like MPLS (multi-protocol label switching) or by optimizing the link metrics used by IP routing protocols such as OSPF (open shortest path first). An example of the result of route optimization for scenarios without or with single link failures is given in Fig. 4, taken from [5].

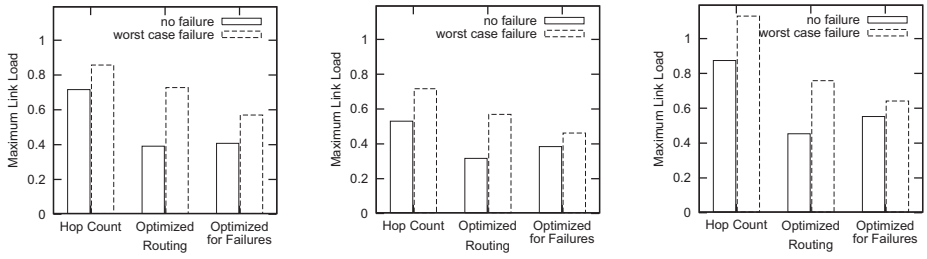


Figure 4: Maximum link loads for normal operation and for worst single link failure with hop-count routing, optimized metrics and metrics optimized for single failure cases for three different networks (for details see [5]).

Networks are interconnected at exchange points and peering points as indicated in Sec. 3. Depending if tier-1 network operators are more interested in delivering best QoS for their customers or in saving money, they have different routing policies to keep packets inside their own network or to route onto other networks as early as possible. Expensive trans-continental links have often been places of network congestion.

5 Admission Control

In circuit switched networks, Admission Control (AC) is implicitly coupled to the act of finding and establishing a circuit through the network. The same concepts can easily be emulated in connection oriented packet networks such as ATM. Connectionless packet switched networks like the Internet or other IP based networks do not maintain per-flow state information in the network nodes. Correspondingly, extra effort is required to integrate admission control functions in IP networks, especially if their inherent resilience mechanisms of fast re-routing after link failures are to be supported.

The following sections list some of the reasons for introducing AC as well as for not introducing AC in IP based networks and summarize possible AC approaches as well as research results comparing the capacity requirements of networks with AC to those of networks operating without AC.

5.1 Motivation for Admission Control

The main reason for introducing Admission Control in IP based networks is to maintain the QoS for already existing traffic flows: If a new traffic flow admitted into a network causes overload on a link, all traffic flows sharing that link will experience packet loss and/or delay. Admission Control protects existing traffic flows from this effect by rejecting a new flow in such a situation. In addition, rejecting such a new flow potentially gives the user or application a better feedback than letting the flow proceed with potentially high levels of packet loss or delay, which translates into QoS degradations for the user.

A major argument for the proponents of AC based capacity management is that although pure dimensioning approaches may be efficient under nominal offered load, substantial additional capacity is required to prevent QoS violations under overload or failure conditions whereas AC approaches simply exhibit increased blocking in those cases.

5.2 Mechanisms

A number of mechanisms have been proposed for Admission Control-like resource management in IP based networks.

Per-Link Admission Control
Concepts like the Bandwidth Broker [6], the Resource Reservation Protocol (RSVP) [7] and Integrated Services [8] get closest to the circuit switching model of having an admission decision for each link in the network. This allows maximum economy of scale for each link but has drawbacks when required to support re-routing for resilience [9]. A more general approach, defining a framework that allows network operators to define what AC mechanism they want to use inside their network domain, is being discussed in the Pre-Congestion Notification (PCN) working group of the Internet Engineering Task Force [10].

Border-to-Border Budgets
Border-to-border budgets realize virtual tunnels for Admission Control. Between every pair of border nodes in a network, there is a certain amount of capacity available that can be assigned to a flow when entering a network. Multi-Protocol Label Switching (MPLS) virtual networks based on IP are an example for such a partitioning of capacity. Although this approach suffers from reduced economy of scale, it has been shown to be the AC method best suited for and most efficient in supporting rerouting and resilient capacity reservation [9]. No state information needs to be transfered in the case of a link failure.

Access-Only AC
Some networks offer sufficient capacity for high-value traffic such as paid video on demand, IPTV or VoIP in the core. In order to support QoS for such applications, an admission control can be realized that takes a number of potential bottleneck links in access networks into account and adds resource management functions for those links to the service control of the supported services. This option obviously requires that access and service provider are the same or that they cooperate sufficiently well to share control.

Prioritization and Deep Packet Inspection
Another approach for supporting a limited set of services with augmented QoS is to simply prioritize packets belonging to those services. In order to limit access to the high-priority class to those services that the network operator deems worth prioritizing, priority marking is either done by mapping virtual channels or using deep packet inspection by marking according to application layer contents in the packets.

All methods share a following fundamental tradeoff between efficiency, signalling and state handling effort and support for resilience: The more efficient a method is, the more state handling and flow rate checking is required in the network and the longer it takes to re-establish reservations after changes in routing, e.g. after a link outage. Furthermore, for some applications (e.g. compressed video conferencing) it is difficult to predict traffic parameters, which requires applications to reserve too much capacity in order to be on the safe side. In addition, there are no realistic inter-domain concepts for admission control that would at the same time support operators' business models and offer reasonable QoS guarantees for dynamic user traffic (see also Sec. 3.3.

5.3 AC and CO on a Single Link

An extensive comparison of Admission Control (AC) and pure Capacity Overprovisioning (CO) approaches has been published in [11].

Define the relative required capacity C_R of a resource management scheme as the ratio $C_R = \frac{C_d}{a \cdot r_m}$ of the required dimensioning capacity C_d to the product of offered load a (in mean number of concurrent flows offered) and the mean data rate r_m per active flow.

Fig. 5 (for details see [11]) depicts the relative required capacity on a single link for AC and CO. It can be seen that both schemes require approximately the same capacity for nominal load. It is also obvious that C_R is very high for very low offered load (no economy of scale).

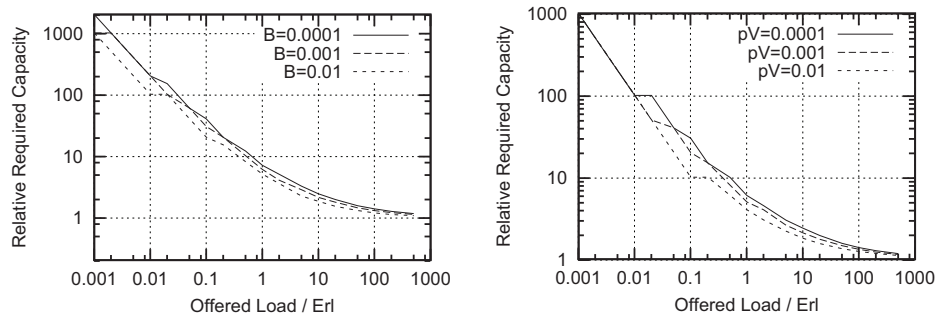


Figure 5: Relative required capacity C_R for admission control (left) and overdimensioning (right) as a function of offered load a (mean number of concurrent flows offered). Parameters: Acceptable blocking probability p_B (left) and acceptable QoS violation probability p_V (right).

Further investigations in [11] confirm the intuitive expectation that if overload scenarios have to be covered, the AC scheme leads to increased blocking under overload whereas CO requires additional capacity proportional to the anticipated overload factor.

5.4 AC and CO in Networks

For applying AC and CO in networks, two results are shown in Fig. 6. The left-hand graph shows a comparison between resilient border-to-border budget based AC and CO in a reference network (for details, see [11]) as a function of offered load whereas the right-hand graph shows the same comparison at a fixed offered load level of $a_{b2b} = 1000$ Erl. Both graphs include options of single or double hot spots with hot spot factor $f_H = 2$.

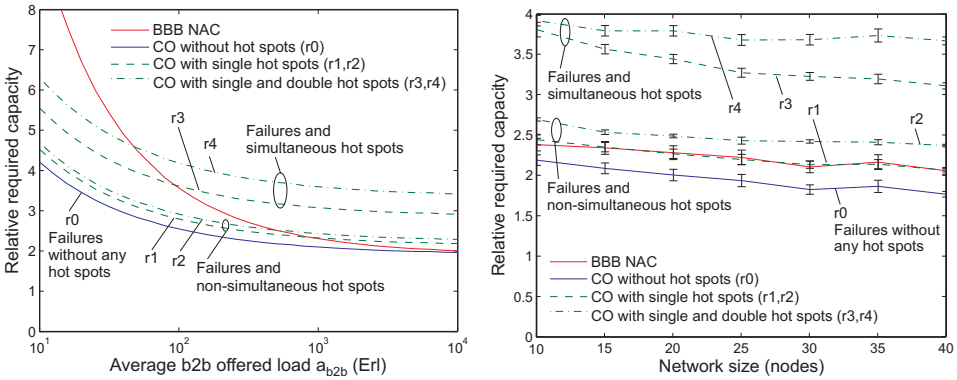


Figure 6: Relative required capacity C_R for resilient admission control and overdimensioning. Left: selected reference network, right: randomly generated networks (see [11]).

Fig. 6 shows the relative required capacity C_R as a function of the mean offered load per border to border relation a_{b2b} for a number of different scenarios. The "BBB NAC" line indicates the relative required capacity for a border-to-border budget based AC scheme that includes capacity for single link failures when a flow is admitted. The overprovisioning lines distinguish between different scenarios: (r0) single link failures but no hot spots, (r1/r3) single link failures or single hot spots with hot spot factor 2 (r1: non simultaneous, r3: also simultaneous), (r2/r4) single link failures and single or double hot spots (r2: non-simultaneous, r4: simultaneous failures and hot spots). It is clearly seen that for low bundle sizes ($a_{b2b} < 50$), AC schemes require more capacity than simple overprovisioning, even

if double hot spots and failures are included in the capacity plans for overprovisioning. In the region of high offered load – when the individual border-to-border budget has sufficient bundle size to exhibit a multiplexing gain comparable to the single link case – AC can save capacity in comparison with CO. At lower offered loads, CO can profit more from economy of scale than AC can.

It can also be seen clearly that the required capacity for CO can be flexibly used to cover temporary overload cases due to hot spots or link failures without rejecting flows. AC, on the other hand, would always retain capacity for link failures and exhibit increased blocking in hot spot scenarios. Only if simultaneous occurrence of overload due to hot spots and due to link failures has to be taken into account (scenarios r3, r4), additional capacity needs to be provided for CO.

6 Conclusions

As application data rates and network capacity are increasing, it is increasingly simple to provide even demanding services with sufficient Quality of Service over the Internet. Traffic profiles of classical Internet applications are changing, making the applications more and more distributed and at the same time less and less suitable for resource management via admission control.

Several options are available for offering admission control on isolated IP networks, whereas dynamic inter-domain QoS is hard to realize for technical as well as business reasons.

Admission control can be a viable option in cases where network and service operators unite to stimulate usage of applications that require more data rate than is usually available for best-effort services. However, this is mostly a transient step only necessary until mean available data rates have caught up and no special precautions are required anymore for those applications. In addition, it was shown that admission control hardly saves any network capacity compared to pure capacity provisioning approaches, provided that low blocking is required under normal operation.

References

[1] findr web site. http://www.forestandthetrees.com/findr/, visited 8. Jul. 2007.

[2] Helsinki Public Transport Map web site. http://transport.wspgroup.fi/hklkartta/, visited 10. Jul. 2007.

[3] J. Charzinski. Observed Performance of Elastic Internet Applications. *Computer Communication*, 26(8):914–925, 2003.

[4] YouTube web site. http://www.youtube.com/, visited 10. Jul. 2007.

[5] U. Walter and J. Charzinski. Optimized Incremental Network Planning. In *Proc. MMB, Nürnberg, Germany*, February 2006.

[6] Z.-L. Zhang, Z. Duan, Y.T. Hou, and L. Gao. Decoupling QoS Control from Core Routers: A Novel Bandwidth Broker Architecture for Scalable Support of Guaranteed Services. In *Proc. ACM SIGCOMM*, pages 71–83, 2000.

[7] R. Braden et al. Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification. IETF RFC 2205, http://www.ietf.org/rfc/rfc2205.txt, 1997.

[8] R. Braden, D. Clark, and S. Shenker. Integrated Services in the Internet Architecture: an Overview. IETF RFC 1633, http://www.ietf.org/rfc/rfc1633.txt, 1994.

[9] M. Menth, S. Kopf, and J. Charzinski. Network Admission Control for Fault-Tolerant QoS Provisioning. In *Proc. HSNMC 2004, Toulouse, France, Springer LNCS Vol. 3079*, June 2004.

[10] Congestion and Pre-Congestion Notification (pcn) working group. http://www.ietf.org/html.charters/pcn-charter.html.

[11] M. Menth, R. Martin, and J. Charzinski. Capacity Overprovisioning for Networks with Resilience Requirements. *ACM SIGCOMM Computer Communication Review (CCR)*, 36(4):87–98, October 2006.

Analysis of a Real-Time Network using Statistical Network Calculus with Effective Bandwidth and Effective Capacity

Kishore Angrishi and Ulrich Killat

Institute for Communication Networks, 4-06,
Hamburg University of Technology,
Hamburg, Germany
{kishore.angrishi,killat}@tu-harburg.de
http://www.tu-harburg.de/et6

Abstract. The Internet traffic consists of an increasing amount of soft real-time flows, which can tolerate very small delay variation and losses. Allowing probabilistic Quality of Service (QoS) violation for these flows can greatly help to improve resource utilization. However, this improvement depends on the statistical properties of competing independent flows and the resources (capacity and buffer) of the multiplexing link. The notion of effective bandwidth summarizes the statistical properties and QoS requirements of the flow. Similarly, the time varying resources availability to the arrival process at the link can be summarized using the notion of effective capacity. This paper uses the concept of effective capacity to express QoS measures at the link using the accurate large deviation technique. Further, it applies the general formulation of effective bandwidth and effective capacity from large deviation theory within the framework of the statistical network calculus for the efficient analysis of the network with finite number of independent sources. This new approach enables efficient utilization of statistical multiplexing of independent flows both at the ingress of the network and along the path of the flow.

Key words: Network calculus, effective bandwidth, effective capacity, Quality-of-Service, statistical multiplexing.

1 Introduction

The increasing demand on transmitting multimedia and other real time applications over the Internet has motivated the study of quality of service (QoS) guarantees. There is a lot of work including an elegant theory under the name of network calculus [6, 7] focusing on analysis of deterministic worst case QoS performance bounds. Network calculus takes an envelope approach to describe arrivals and services in a network. The strength of the deterministic network calculus is that it can be used to determine the delay and backlog over multiple network nodes. However, the worst case deterministic performance bounds

are often conservative and far away from practical use since they lead to low utilization of network resources. This motivates the probabilistic extension of the network calculus, commonly referred to as "statistical network calculus". The significant steps towards statistical network calculus are presented in [8, 9, 2, 16], where the concept of effective envelopes and effective service envelopes are derived to describe arrivals and services in a network respectively. The main advantage of statistical network calculus over its deterministic counterpart is its capability to utilize statistical multiplexing within a framework to profit from multiplexing gain. However this utilization of statistical multiplexing gain is limited to network ingress, but there are still possibilities of gain due to statistical multiplexing of the independent flows along the path of the flow [4, 17].

The related analytical technique considering statistical multiplexing of arrival flows are well understood for single server under the theory of large deviations. Probably the most influential framework for service provisioning derived from the theory of large deviations is the effective bandwidth [13], which describes the minimum bandwidth required to provide an expected service for a given traffic. The effective bandwidth of a flow determines the bandwidth somewhere between the average and peak rate of the flow. Similarly, we introduce the concept of effective capacity to represent the stochastic service received by the arriving traffic at a node.

In this paper, we integrate the concept of effective bandwidth and effective capacity into the formalism of the statistical network calculus. As a result, it is feasible to effectively utilize statistical multiplexing gain among independent flows. The relationship between network calculus and effective bandwidth has been first investigated by Chang [6]. Later a formal relationship between effective bandwidth and statistical network calculus was proposed in [5]. This relationship was further improved in [4], which together with (approximate) invariance property of effective bandwidth [14, 12] was used within the framework of statistical network calculus to analyze feed forward network. Though this approach enables efficient usage of statistical multiplexing gain among independent flows, it requires a minimum number of independent flows (effective threshold) to be multiplexed at each node.

In [17], the relationship proposed by Chang was extended and moment generating functions were used to calculate the end-to-end performance bounds in a network. This paper continues to explore this relationship exploiting the accurate large deviation estimate of loss asymptotic studied in [10, 1] and use effective bandwidth to describe the stochastic arrival process inside the network. Further, we introduce the analogous concept of effective capacity to represent stochastic services in a network within the framework of statistical network calculus, which enables us to efficiently analyze a feed forward network. The framework proposed is useful for end-to-end performance bounds analysis in a multi-node network and for efficient utilization of statistical multiplexing gain among independent flows.

The paper is organized as follows. In section 2, we introduce the concept of effective capacity within large deviations theory and provide QoS measure at

single server in terms of effective capacity for constant bit rate as well as variable bit rate traffic. Section 3 provides a brief introduction of network calculus and statistical network calculus, and then proceeds to present our concise framework of statistical network calculus with effective bandwidth and effective capacity to analyze a feed forward network with multiple servers. the numerical results are shown in section 4.

2 Effective Capacity in Large Deviations Theory

The goal of this section is to introduce the concept of effective capacity in Large Deviations Theory (LDT) under many sources asymptotic regime. In many sources asymptotic regime, as the number of traffic sources N $\rightarrow \infty$, the service rate Nc and buffer size Nb increase linearly with N. Analogous to effective bandwidth being the statistical descriptor of an arrival process, here we show that the effective capacity can be used to represent the stochastic service received by the arriving traffic in a single server. Stochastic services are commonly observed in wireless network and in differentiated services-capable network.

We first introduce necessary notation, and then present our results of QoS measure for single server in terms of effective capacity and effective bandwidth.

Consider a finite capacity buffer (queue) which is accessed by a number of independent, identical, stationary and ergodic sources. We assume that the time is discrete or slotted. At any discrete time instant, denoted by n, a finite number of cells A_n is emitted from the active sources into the buffer which has finite waiting room B and the buffer is drained at stochastic rate of S_n cells per unit time. If there is insufficient space in the buffer then the corresponding cells which cannot be admitted are lost. We consider a stochastic service process which is stationary and ergodic with $E[S_n] > E[A_n]$. In the case when $B = \infty$, this condition guarantees the existence of a stationary workload or queue length process. When $B < \infty$, it is not necessary for the existence of a stationary workload process but given that we are interested in rare buffer overflows then this condition is necessary. The particular case of the situation we consider is when S_n is the service given to the aggregate traffic from a large number of sources, say N. This N will act as a scaling parameter on the system and will be a measure of the size. It is also assumed the buffer size is $B = N \cdot b$. Thus the parameter $(\frac{S_n}{N}, b)$ can be thought as the nominal capacity and buffer space if only one source was accessing the system.

Consider N independent sources, each with constant input rate A_n with $E\left[\frac{S_n}{N}\right] > E[A_n] > 0$. In practical applications the support of the distribution of S_n is in [0,C] and the distribution of A_n is in [0,K]. The bound C is the maximum service rate of a link and the bound K is referred to as peak rate of the arrival from a source with $min(\frac{S_n}{N}) < E[A_n]$ or $E\left[\frac{S_n}{N}\right] < K$.

The total amount of work offered during the time interval [0,t) from a source i is $X_{t,i} = \sum_{j=0}^{t-1} A_{j,i}$, where $A_{j,i}$ is the number of cells emitted from source i at a discrete time instant j. Similarly the amount of work offered from all sources is $X_t^{(N)} = \sum_{i=0}^{N} X_{t,i}$.

The average amount of service offered to the traffic from a source during time interval [0,t) is $Y_t = \frac{1}{N}\sum_{j=0}^{t-1} S_j$ and the amount of service offered to all sources is $Y_t^{(N)} = N \cdot Y_t = \sum_{j=0}^{t-1} S_j$. It is assumed that the moment generation functions of $Y_t^{(N)}$, $X_t^{(N)}$

$$M_s(\theta, t) = E\left[e^{\theta \cdot Y_t^{(N)}}\right] \tag{1}$$

$$M_a(\theta, t) = E\left[e^{\theta \cdot X_t^{(N)}}\right] = \prod_{i=1}^{N} E\left[e^{\theta \cdot X_{t,i}}\right] \tag{2}$$

and the log-moment generation functions of $Y_t^{(N)}$, $X_t^{(N)}$

$$\varphi(\theta, t) = \frac{1}{t} log E\left[e^{\theta \cdot Y_t^{(N)}}\right] \tag{3}$$

$$\varphi^a(\theta, t) = \frac{1}{t} log E\left[e^{\theta \cdot X_t^{(N)}}\right] = \sum_{i=1}^{N} \frac{1}{t} log E\left[e^{\theta \cdot X_{t,i}}\right] = \sum_{i=1}^{N} \varphi_i^a(\theta, t) \tag{4}$$

exist for all $\theta > 0$ and $t > 0$. Then *effective bandwidth* function of $X_t^{(N)}$ is given by

$$\alpha(\theta, t) = \frac{1}{\theta t} log E\left[e^{\theta \cdot X_t^{(N)}}\right] = \sum_{i=1}^{N} \frac{1}{\theta t} log E\left[e^{\theta \cdot X_{t,i}}\right] = \sum_{i=1}^{N} \alpha_i(\theta, t) \tag{5}$$

Similarly, the asymptotic log-moment generation functions and the corresponding asymptotic effective bandwidth functions are given by

$$\varphi(\theta) = \lim_{t\to\infty} \frac{1}{t} log E\left[e^{\theta \cdot Y_t^{(N)}}\right], \ \varphi^a(\theta) = \lim_{t\to\infty} \frac{1}{t} log E\left[e^{\theta \cdot X_t^{(N)}}\right] \tag{6}$$

$$\alpha(\theta) = \lim_{t\to\infty} \frac{1}{\theta t} log E\left[e^{\theta \cdot X_t^{(N)}}\right] \tag{7}$$

Note by assumption this exists for all finite θ since the moment generation function exist for all t and $\theta > 0$.

The following property is needed for the results in the rest of this paper.

Property 1 *The asymptotic log-moment generation functions defined in eq.(6) (i) are finite for all $\theta \in R$. (ii) are differentiable for all $\theta \in R$.*

We first begin by obtaining the estimate for probability of exceedance of the buffer level Nb for the queuing model given above assuming that there is unlimited waiting room.

Consider the discrete time queuing system with stochastic output rate $S_t^{(N)}$ at time t and with unlimited buffer.

Let $W_t^{(N)}$ denote queue content (workload) at time t. Then $W_t^{(N)}$ is given by:

$$W_t^{(N)} = max(0, W_{t-1}^{(N)} + A_t^{(N)} - S_t^{(N)}) \tag{8}$$

The stationary workload $W^{(N)}$ is then given by:

$$W^{(N)} = \sup_{t\in\{0,1,2,\ldots\}} (X_{-t}^{(N)} - Y_{-t}^{(N)}) \tag{9}$$

where $X_{-t}^{(N)}$ denotes the total amount of arriving cells and $Y_{-t}^{(N)}$ denotes the total amount of arriving cells served at the link in the interval (-t, 0].

We now introduce the concept of effective capacity using the queue model described above, with large N number of sources, each having constant bit-rate of λ.

The queue content (workload) at time t $W_t^{(N)}$ and stationary workload $W^{(N)}$ given in eq.8, eq.9 become,

$$W_t^{(N)} = max(0, W_{t-1}^{(N)} + N \cdot \lambda - S_t^{(N)}) \tag{10}$$

$$W^{(N)} = \sup_{t \in \{0,1,2,\ldots\}} (N \cdot \lambda \cdot t - Y_{-t}^{(N)}) \tag{11}$$

The typical application of LDT for queuing systems concerns the probability that the current buffer occupancy W_t exceeds a certain buffer occupancy value B ($B = Nb$). The probability drops negative exponentially with increasing buffer occupancy B [6], i.e.

$$P(W_t > B) \approx Ae^{-\theta B} \tag{12}$$

with asymptotic decay rate θ, where A is asymptotic constant. The definition of effective capacity is closely related to the exponential decay rate. Effective capacity describes the maximum constant arrival rate of the traffic to a node with a given stochastic service rate to provide an expected QoS. The effective capacity of a server determines the constant arrival rate of the traffic somewhere between the average and minimum service rate of the server. Then, the *effective capacity* function of service process $Y_t^{(N)}$ based on many sources asymptotic ($N \to \infty$) is defined as

$$\beta(\theta, t) = -\frac{1}{\theta t} logE\left[e^{-\theta \cdot Y_t^{(N)}}\right] \tag{13}$$

where θ, t are the parameters which are defined by the context of the multiplexing link, i.e., the characteristics of the stochastic service, the QoS it can offer. The time parameter t corresponds to the most probable duration of the buffer busy period prior to overflow.

The *effective capacity* based on the large buffer asymptotic ($B \to \infty$) is defined as

$$\beta(\theta) = \lim_{t \to \infty} -\frac{1}{\theta t} logE\left[e^{-\theta \cdot Y_t^{(N)}}\right] \tag{14}$$

Observe that (14) is a special case of (13) for t $\to \infty$. Indeed, the effective capacity formula (14) gives an accurate measure of resource usage when the link buffer is large, in which case the time parameter t (which is related to the time for buffer overflow) becomes large.

The buffer overflow probability given by (12) is found to be too conservative when $A = 1$ and a convincing argument is made in [3], that the asymptotic constant A may be in fact be small or large, reflecting the extent of the statistical multiplexing gain.

The following result characterizes the buffer overflow probability in the form (12) for large N number of sources, each having constant arrival rate of λ. This follows from direct application of Bahadur-Rao theorem [1]

Lemma 1 *For the model described by eqs. (8)-(13) and with assumption that arrival process and service process at a given node satisfies the property 1,* $N \to \infty$

$$P\left\{W^{(N)} > Nb\right\} \approx \frac{e^{-NI(\lambda,b,-\theta)}}{\sqrt{4\pi NI(\lambda,b,-\theta)}} \tag{15}$$

where

$$NI(\lambda,b,-\theta) = \inf_t \left[-N\theta(\lambda t - b) + \theta t\beta(\theta,t)\right] \tag{16}$$

where θ is the unique solution of

$$N(\lambda t - b) = \frac{M_s^{'}(-\theta,t)}{M_s(-\theta,t)} \tag{17}$$

For a proof of Lemma 1, see the Appendix.

We now state the following result which characterizes QoS measures for the case where the sources generate traffic at variable bit-rates (VBR) with effective bandwidth function $\alpha_i(\theta,t)$.

Lemma 2 *With assumption that arrival process and service process at a given node satisfies the property 1,* $N\to\infty$*, the buffer overflow probability of a queuing model given by eqs. (8)-(13) with server having effective capacity function of* $\beta(\theta,t)$ *serving the traffic from large N number of sources, each having effective bandwidth function of* $\alpha_i(\theta,t)$ *and satisfying the condition* $\sum_{i=1}^{N}\alpha_i(\theta,t) \leq \beta(\theta,t)$ *is given as,*

$$P\left\{W^{(N)} > Nb\right\} \approx \frac{e^{-NI(b)}}{\sqrt{4\pi NI(b)}} \tag{18}$$

where

$$NI(b) = \inf_t \left[N\theta b - \theta t\left[\alpha(\theta,t) - \beta(\theta,t)\right]\right] \tag{19}$$

where θ is the unique solution of

$$Nb = \frac{M_a^{'}(\theta,t)}{M_a(\theta,t)} - \frac{M_s^{'}(-\theta,t)}{M_s(-\theta,t)} \tag{20}$$

For a proof of Lemma 2, see the Appendix.

In spite of the successes in analyzing the single node case in both the large buffer and many sources contexts, there has been only limited success at identifying the asymptotic for network models. However, there has been some partial success in the many sources case. This is due to the fact that in the many sources context, the resulting measure is a convolution and the buffer occupancy probability goes to zero even for very small buffers. This result has been exploited by Wischik [12] who has shown that the moment generating function (m.g.f.) for a single input does not change as it passes through a node when multiplexed with many similar inputs. Since the m.g.f. is the necessary information required to determine the overflow asymptotic, one can analyze large in-tree networks where each node receives only a small number of inputs from a large number of other independent nodes. Though the result is obtained using many sources asymptotic analysis, this holds already for a surprisingly small number of competing

flows even in the presence of aggressive TCP traffic [14]. The issue is to find the minimum number of sources (effective threshold) required to observe this invariance property. An alternative method to determine resource requirements of finite number of traffic flows in a packet network is the network calculus. In [4], the (approximate) invariance property of effective bandwidth was used within the framework of statistical network calculus to efficiently utilize statistical multiplexing gain between independent flows. However, this approach requires a minimum number of independent flows, defined by effective threshold, to be multiplexed at each node inside the network. Another approach would be to identify the effective bandwidth of the output traffic from a node. In [6, 17], the statistical network calculus using moment generating functions was used to calculate the end-to-end performance bounds in a network, which can be used to describe the effective bandwidth of the output traffic of a single node. In the next section, we extend the results in [6, 17], using the effective bandwidth and the effective capacity as stochastic descriptors of the arrival and the services process in a network, within the framework of statistical network calculus.

3 Statistical Network Calculus with Effective Bandwidth and Effective Capacity

In network calculus [6, 7] flow's arrival, flow's departure and the service it receives at a node are described by real valued cumulative functions $A(t)$, $B(t)$, and $S(t)$ [6, 7]) respectively in an interval $(0, t]$. We assume that there are no arrivals in the interval (-∞, 0]. Clearly $A(t)$, $B(t)$ and $S(t)$ are non-negative and non-decreasing in t. The foundations of network calculus are min-plus convolution and min-plus deconvolution. The operations of min-plus convolution ($\otimes$) and deconvolution ($\oslash$) of the arrival process $A(t)$ and service process $S(t)$ [6] are defined for any $t, \tau \geq 0$ as, [7]

$$(A \otimes S)(t) = \inf_{\tau \in [0,t]} [A(t-\tau) + S(\tau)] \leq B(t) \tag{21}$$

$$(A \oslash S)(t) = \sup_{\tau \in [0,\infty]} [A(t+\tau) - S(\tau)] \geq B(t) \tag{22}$$

The applicability of network calculus suffers from the worst-case modeling that is used. Since traffic is statistically multiplexed at network nodes, such scenarios are extremely rare. Therefore, a probabilistic view, which considers that traffic in a packet network employing statistical multiplexing increases the achievable utilization in the network by tolerating rare adversarial events. The statistical network calculus seeks to quantify the statistical multiplexing gain while maintaining the algebraic aspects of the deterministic calculus. The particular strength of network calculus is the efficient analysis of servers in series. Tandem servers can be concatenated using min-plus convolution and performance bounds can be derived based on min-plus deconvolution. However such concatenation of tandem servers using min-plus convolution in statistical network calculus leads to deterioration of performance bounds in the number of servers in series. Statistical network calculus has been a hot topic recently and

several groups have produced new results especially, [16]. However, it can be shown that these results are directly related to the fundamental results presented in [8] and [9].

We follow the framework for a statistical network calculus presented in [8] and [9]. For traffic arrivals, we use a probabilistic measure called effective envelope [8]. An effective envelope for an arrival process A is defined as a non-negative function $\mathcal{G}^\varepsilon$ such that for all $t \geq 0$

$$P\{A(t+\tau) - A(t) \leq \mathcal{G}^\varepsilon(\tau)\} \geq 1 - \varepsilon \tag{23}$$

In other words, an effective envelope provides a stationary bound for an arrival process. Effective envelopes can be obtained for individual flows, as well as for multiplexed arrivals [5]. To characterize the available service to a flow or a collection of flows we use effective service curves [9] which can be seen as a probabilistic measure of the available service. Given an arrival process A, an effective service curve is a non-negative function $\mathcal{S}^\varepsilon$ that satisfies for all $t \geq 0$

$$P\{B(t) \geq A \otimes \mathcal{S}^\varepsilon(t)\} \geq 1 - \varepsilon \tag{24}$$

By letting $\varepsilon \to 0$ in eq.(23) and eq.(24), we recover the arrival envelopes and service curves of the deterministic calculus with probability one. The statistical network calculus has also been related to other analytical techniques. Here, we integrate the concept of effective bandwidth and effective capacity into the formalism of the statistical network calculus. There have been already some efforts to express general formulation of effective bandwidth within the framework of statistical network calculus [5, 4]. The term "effective envelope" as introduced in [8] suggests a connection to the notion of effective bandwidth, but without making that connection explicit. In [5] the authors establish a formal relationship between the two concepts, and thus, link the effective bandwidth theory to the statistical network calculus. In [5] the authors use this relationship to construct an effective envelope for a traffic class if its effective bandwidth is known. Once the effective envelope is fixed at the network ingress, it is used to represent the traffic along the network. This prohibits any consideration of statistical multiplexing along the path of traffic. Moreover the traffic's effective envelope deteriorates along its path by a right-shift of the envelope with the amount (h-1)d [5], where h is number of nodes along the path and d is the a priori delay threshold at each node. In our approach, we propose to use the traffic's effective bandwidth to describe the arrival process in the network and the server's effective capacity function to describe the service process in the node. At each network node, the effective envelope is constructed from the effective bandwidth of incoming traffic and the effective service envelope is constructed from the effective capacity function of the server to apply results from the statistical network calculus. To construct the effective envelope from the effective bandwidth, the relationship presented in [4] is used. In [4], the formal relationship defined in [5] has been improved using the accurate large deviation estimate of loss asymptotic studied in [10, 1]. Given an arrival process A with effective

bandwidth α, the effective envelope $\mathcal{G}^\varepsilon$ of the arrival process for any $t \geq 0$ and $\theta \geq 0$ is given by [4]

$$\mathcal{G}^\varepsilon(t) = \inf_\theta \left\{ t\alpha(\theta, t) - \frac{log\varepsilon'}{\theta} \right\} \tag{25}$$

where

$$log\varepsilon' = log\varepsilon + \frac{\frac{1}{2}log(-4\pi log\varepsilon)}{1 - \frac{1}{2log\varepsilon}}$$

To derive the effective service envelope from the effective capacity function of the server, following result can be used.

Lemma 3 *Given a service process S with the effective capacity β, the effective service envelope $\mathcal{S}^\varepsilon$ of the service process for any $t \geq 0$ and $\theta \geq 0$ is given by*

$$\mathcal{S}^\varepsilon(t) = \inf_\theta \left\{ \frac{log\varepsilon'}{\theta} + t\beta(\theta, t) \right\} \tag{26}$$

where

$$log\varepsilon' = log\varepsilon + \frac{\frac{1}{2}log(-4\pi log\varepsilon)}{1 - \frac{1}{2log\varepsilon}}$$

The proof is analogous to the proof found in the Appendix of [4], with θ being replaced by $-\theta$.

The following lemmas are necessary for finding effective capacity of the network of servers in series and the performance bounds in a server or network of servers.

Lemma 4 *Let $S^1(t)$ and $S^2(t)$ be two independent random service processes of two servers in series and $\beta^1(\theta, t)$, $\beta^2(\theta, t)$ be their corresponding effective capacity functions for any $t \geq 0$ and $\theta \geq 0$. Then the effective capacity function of the network of two servers in series satisfies*

$$\beta^{(S^1 \otimes S^2)(t)}(\theta, t) \geq -\frac{log(t+1)}{\theta t} + \frac{1}{t}\left[(t\beta^1(\theta) \otimes t\beta^2(\theta))(t)\right] \tag{27}$$

For a proof of Lemma 4, see the Appendix.

Lemma 5 *Let $A(t)$ and $S(t)$ be two independent random arrival and service processes at a node and $\alpha(\theta, t)$, $\beta(\theta, t)$ be their corresponding effective bandwidth and effective capacity functions for any $t \geq 0$ and $\theta \geq 0$. Then the effective bandwidth function of the output process from the node satisfying the condition $\alpha(\theta) < \beta(\theta)$ for any $\theta \geq 0$ satisfies*

$$\alpha^{(A \oslash S)(t)}(\theta, t) \leq \alpha(\theta) - \frac{1}{\theta t} log[1 - e^{\theta(\alpha(\theta) - \beta(\theta))}] \tag{28}$$

For a proof of Lemma 5, see the Appendix. The resulting effective bandwidth function of output process remains finite, if the condition for stability $\alpha(\theta) < \beta(\theta)$ is fulfilled for all $\theta \geq 0$. Lemma 4 together with Lemma 3 is useful for constructing a statistical network service curve from the effective capacity function of the network of servers in series. Lemma 5 together with the eq.25 [4]

is useful in deriving a probabilistic upper bound on $(A \oslash S)(t)$ which is violated at most with the probability ϵ and in turn establishes probabilistic backlog, delay, and output bounds. The framework presented here enables efficient utilization of statistical multiplexing of independent traffic flows both at the ingress of the network and along the path of the traffic and also simplifies the construction of statistical network service envelope.

4 Numerical Experiment

In this section, we present a simple numerical example of the tandem nodes with cross traffic shown in Fig.1 to illustrate the multiplexing gain attained using our framework of statistical network calculus with effective bandwidth and effective capacity. Let N be the number of through-flows, M_1 and M_2 cross-flows at first node and second node respectively. We use stationary, single token bucket regulated traffic sources with maximum burst size b, rate r and peak rate P. For the effective bandwidth of such regulated source i, see for example [5], $\alpha_i(\theta, t) \leq \frac{1}{\theta t} log\left(1 + \frac{rt}{[rt+b,Pt]^-}(e^{\theta[b+rt,pt]^-} - 1)\right)$, where $[x, y]^-$ means minimum of (x, y). The effective bandwidth of the aggregate of independent flows is found using eq.(5). The servers each have capacity C. The minimum service seen by the through-flows is determined under the general scheduling model [6]. For example, if the effective bandwidth of cross-flows M is $\alpha_M(\theta, t)$, the minimum effective capacity available to the through-flows N is given by Ct - $\alpha_M(\theta, t)$. To make a comparison of the conventional statistical network calculus presented in [9, 5] and approach using (approximate) invariance of effective bandwidth [4], we perform the analysis with the same model as in [4]. The effective threshold for this model was found to be 15 flows using a model simulation setup for any $1 \geq \theta \geq 0$. Hence we set cross-flows $M_1 = 15$ (at node 1), $M_2 = 15$ (at node 2), so that number of independent flows multiplexing at each node is greater than effective threshold. There are $(N + M_1)$ independent incoming flows to node 1, and we increase the through-flows N to a maximum feasible value. All flows have the same individual deterministic (b, r, P) arrival curve with $b = 500byte, r = 200byte/ms, P = 1000byte/ms$. Node 1 and node 2 are both served at $12500byte/ms$. Finally, we set the expected delay bound to $d = 2ms$ at every node. Fig.2 shows the maximum number of soft real-time through-flows using statistical network calculus results found in [5], approach found in [4] and using our approach for the demo system under different violation probabilities ε. In our approach, we use *lemmas 3,5*, eqs. (25) (5) and results of statistical network calculus found in [5] to find the maximum possible number of through-flows N in the demo system.

In summary, from Fig.2 it is observed that using the arrival curves from network calculus (Det Env) enables 22 through-flows, the effective envelope from [5] (Eff Env) enables 35 to 42 through-flows, and effective envelope using Bahadur-Rao improvement from eq.(25) (Eff Env B-R) enables 36 to 45 through-flows respectively. Clearly, Bahadur-Rao improvement provides a better performance than the existing approaches. If *(approximate) invariance* property of effective bandwidth is considered, effective envelope (Inv Eff Env [Chernoff]) and effective

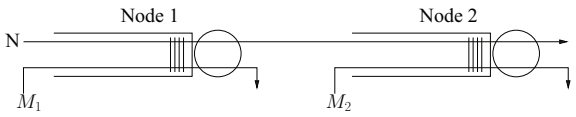


Fig. 1. Tandem nodes with cross-traffic

envelope (Inv Eff Env [B-R]) using Bahadur-Rao improvement enable 47 to 57 and 49 to 61 through-flows respectively. Using our statistical network calculus framework with effective bandwidth and effective capacity, 45 to 55 through-flows can be allowed using Chernoff's bound (Eff Env [Chernoff] - EB-EC method). While 47 to 59 flows can be allowed using Bahadur-Rao improvement (Eff Env [B-R]- EB-EC method). It can be clearly seen that there is efficient usage of statistical multiplexing gain in our approach. The approach using (approximate) invariance of effective bandwidth provides the best results and upper bounds the maximum number of through-flows that can be allowed. However, by exploiting the accurate large deviations results within our new framework using effective bandwidth and effective capacity, we get comparable results. It can be seen that improvement is achieved by exploiting the statistical multiplexing between through-flows and cross-flows at both nodes, whereas, the conventional approach considers the statistical multiplexing gain only at the ingress node.

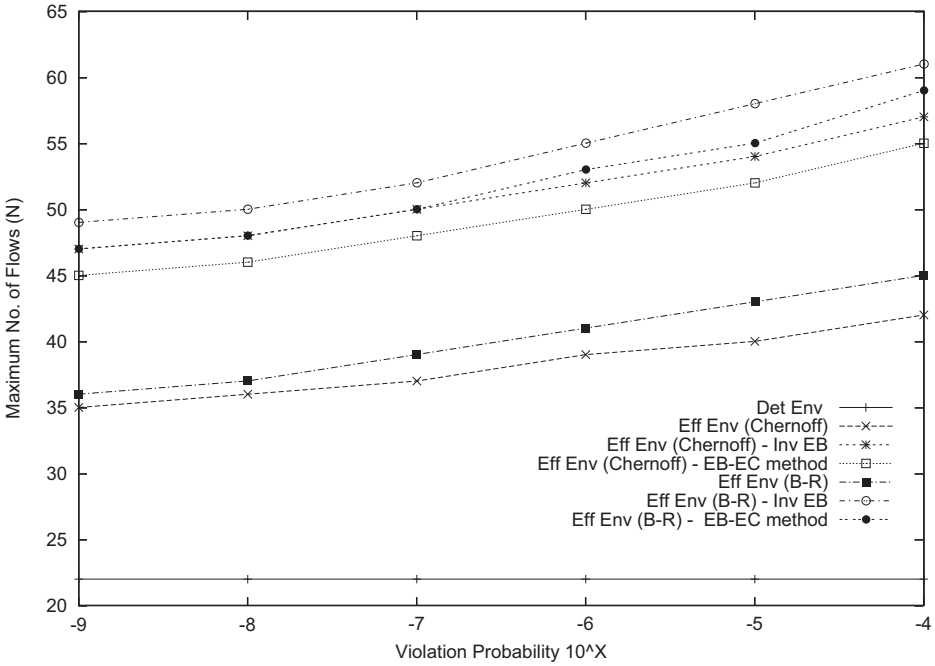


Fig. 2. Utilization ($M_1 = 15$, $M_2 = 15$)

5 Conclusion

In this paper, we introduced the concept of effective capacity in large deviations theory under many source asymptotic regime and derived QoS measures at a node in-terms of effective capacity for constant bit rate as well as for variable bit rate traffic with a given effective bandwidth function. We then presented an end-to-end probabilistic network calculus with effective bandwidth and effective capacity. This results in an efficient use of the network resources and an increase in the number of traffic flows that can be accommodated in the network without violating their soft real-time QoS constraints.

References

1. Bahadur, R., R. and Ranga Rao, R.,: On deviations of the sample mean. Ann. Math. Statist. 31 1960.
2. Cruz, R. L. ,: Quality of service management in integrated services networks. In Proc. the 1st Semi-Annual Research Review, CWC, USCD, 1996.
3. Choudhury, G. L., Lucantoni, D. M., and Whitt, W. ,: Squeezing the most out of ATM, IEEE Trans. Commun., vol. 44, pg. 203-217, Feb. 1996.
4. Angrishi, K., Zhang, S., Killat, U.,: Analysis of a Real-Time Network using Statistical Network Calculus with Approximate Invariance of Effective Bandwidth, 15.ITG/GI - Fachtagung Kommunikation in Verteilten Systemen (KiVS 2007), Bern, Switzerland, March 2007
5. Li, C., Burchard, A., and Liebeherr, J.,: A Network Calculus with Effective Bandwidth. Technical Report CS-2003-20, University of Virginia, 2003.
6. Chang, C.-S.,: Performance Guarantees in Communication Networks. Springer-Verlag, 2000.
7. Le Boudec J.-Y., and Thiran P.,: Network Calculus A Theory of Deterministic Queuing Systems for the Internet, ser. LNCS. Springer-Verlag, 2001, no. 2050.
8. Boorstyn, R.-R., Burchard, A., Liebeherr, J., and Oottamakorn C.,: Statistical Service Assurances for Traffic Scheduling Algorithms, IEEE Journal on Selected Areas in Communications, 18(12):2651-2664, 2000.
9. Burchard, A., Liebeherr, J., and Patek, S. D.,: A calculus for end-to-end statistical service guarantees (revised). Technical Report CS-2001-19, University of Virginia, Computer Science Department, May 2002.
10. Likhanov, N., and Mazumdar, R. R.,: Cell loss asymptotics for buffers fed with a large number of independent stationary sources, Journal of Applied Probability, 1999.
11. Chang, C.-S., and Thomas, J.A.,: Effective bandwidth in high speed digital networks, IEEE Journal on Selected Areas in Communications, vol. 13, no. 6, 1995.
12. Wischik, D.,: The output of a switch, or, effective bandwidths for networks, Queueing Systems, Volume 32, 1999.
13. Kelly,F. P.,: Notes on effective bandwidths, ser. Royal Statistical Society Lecture Notes. Oxford University, 1996, no. 4.
14. Abendroth, D., and Killat, U.,: An advanced traffic engineering approach based on the approximate invariance of effective bandwidths, Telecommunication Systems Journal, Kluwer Publishers, 2004, vol 27 (2-4).
15. Montgomery, M., and de Veciana, G.,: On the relevance of time scales in performance oriented traffic characterizations. In Proc. of IEEE INFOCOM'96, pp. 513-520, April 1996.

16. Ciucu, F., Burchard, A., and Liebeherr, J.,: A network service curve approach for the stochastic analysis of networks, in Proc. ACM SIGMETRICS, June 2005, 279-290.
17. Fidler, M.,: An End-to-End Probabilistic Network Calculus with Moment Generating Functions. Proceedings of IWQoS, June 2006.

Appendix

Proof of Lemma 1

From the definition of $W^{(N)}$ and noting from the stationary increment property of $Y_t^{(N)}$ that $Y_{-t}^{(N)}$ has the same distribution as $Y_t^{(N)}$. Pick a θ_1 such that - $\frac{1}{N}\varphi(-\theta_1) - \lambda\theta_1 > 2\epsilon$. Then since $\varphi(\theta_1, t) \to \varphi(\theta_1)$, there must be a t_1 such that for all $t > t_1$, both

$$-\frac{1}{N}\varphi(-\theta_1, t) - \lambda\theta_1 > \epsilon$$

and

$$\theta_1 b + t\epsilon > \sup_\theta[-\frac{1}{N}\varphi(-\theta, 1) - \theta\lambda]$$

$$\begin{aligned}
P\left\{W^{(N)} > Nb\right\} &\leq \sum_{t=1}^{\infty} P\left\{X_t^{(N)} > Y_t^{(N)} + Nb\right\} \\
&= \sum_{t=1}^{\infty} P\left\{-Y_t^{(N)} > -X_t^{(N)} + Nb\right\} \\
&= \sum_{t=1}^{t_1-1} P\left\{-Y_t^{(N)} > -X_t^{(N)} + Nb\right\} + \sum_{t=t_1}^{\infty} P\left\{-Y_t^{(N)} > -X_t^{(N)} + Nb\right\}
\end{aligned}$$

Applying the Bahadur-Rao theorem [1] to first term and Chernoff bound to second term respectively in the summation above, we obtain:

$$\begin{aligned}
&\leq \sum_{t=1}^{t_1-1} \frac{e^{-N\sup_\theta[-\theta(\lambda t-b)-\frac{t}{N}\varphi(-\theta,t)]}}{\sqrt{2\pi\theta^2\sigma^2 N}} + \sum_{t=t_1}^{\infty} e^{-N[\theta_1 b + t\{-\frac{1}{N}\varphi(-\theta_1,t)-\lambda\theta_1\}]} \\
&\leq (t_1-1)\frac{e^{-N\min_{t<t_1}\sup_\theta[-\theta(\lambda t-b)+\frac{\theta t}{N}\beta(\theta,t)]}}{\sqrt{2\pi\theta^2\sigma^2 N}} + \sum_{t=t_1}^{\infty} e^{-N[\theta_1 b+t\epsilon]} \\
&\leq (t_1-1)\frac{e^{-N\min_{t<t_1}\sup_\theta[-\theta(\lambda t-b)+\frac{\theta t}{N}\beta(\theta,t)]}}{\sqrt{2\pi\theta^2\sigma^2 N}} + \frac{e^{-N[\theta_1 b+t_1\epsilon]}}{1-e^{-N\epsilon}}
\end{aligned}$$

Since the second term of the expression above is a fixed value and for large N we get,

$$P\left\{W^{(N)} > Nb\right\} \approx \frac{e^{-NI(\lambda,b,-\theta)}}{\sqrt{2\pi\theta^2\sigma^2 N}} \tag{29}$$

The term $N\sigma^2\theta^2$ can be approximated by $2NI(\lambda, b, -\theta)$ [15]. thereby resulting in the following approximation from eq.(29):

$$P\left\{W^{(N)} > Nb\right\} \approx \frac{e^{-NI(\lambda,b,-\theta)}}{\sqrt{4\pi NI(\lambda,b,-\theta)}} \tag{30}$$

Hence yielding the desired result.

Proof of Lemma 2

The proof is similar to that of Lemma 1.

From the definition of $W^{(N)}$ and noting from the stationary increment property of $X_t^{(N)}$, $Y_t^{(N)}$ that $X_{-t}^{(N)}$ and $Y_{-t}^{(N)}$ has the same distribution as $X_t^{(N)}$ and $Y_t^{(N)}$ respectively. Pick a θ_1 such that - $\frac{1}{N}\varphi(-\theta_1) - \varphi_1^a(\theta) > 2\epsilon$. Then since $\varphi(\theta_1, t) \to \varphi(\theta_1)$ and $\varphi^a(\theta_1, t) \to \varphi^a(\theta_1)$ there must be a t_1 such that for all $t > t_1$, both

$$-\frac{1}{N}\varphi(-\theta_1, t) - \varphi_1^a(\theta_1, t) > \epsilon$$

and

$$\theta_1 b + t\epsilon > \sup_\theta[-\frac{1}{N}\varphi(-\theta, 1) - \varphi_1^a(\theta_1, 1)]$$

$$\begin{aligned}
P\left\{W^{(N)} > Nb\right\} &\leq \sum_{t=1}^{\infty} P\left\{X_t^{(N)} - Y_t^{(N)} > Nb\right\} \\
&= \sum_{t=1}^{\infty} P\left\{X_t^{(N)} - Y_t^{(N)} > Nb\right\} \\
&= \sum_{t=1}^{t_1-1} P\left\{X_t^{(N)} - Y_t^{(N)} > Nb\right\} + \sum_{t=t_1}^{\infty} P\left\{X_t^{(N)} - Y_t^{(N)} > Nb\right\}
\end{aligned}$$

Applying the Bahadur-Rao theorem [1] to first term and Chernoff bound to second term respectively in the summation above, we obtain:

$$\begin{aligned}
&\leq \sum_{t=1}^{t_1-1} \frac{e^{-N\sup_\theta[\theta b - [\varphi_1^a(\theta,t) + \frac{t}{N}\varphi(-\theta,t)]]}}{\sqrt{2\pi\theta^2\sigma^2 N}} + \sum_{t=t_1}^{\infty} e^{-N[\theta_1 b + t\{-\frac{1}{N}\varphi(-\theta_1,t) - \varphi_1^a(\theta_1,t)]} \\
&\leq (t_1 - 1)\frac{e^{-N\min_{t<t_1}\sup_\theta[\theta b - \theta t[\alpha_1(\theta,t) - \frac{1}{N}\beta(\theta,t)]]}}{\sqrt{2\pi\theta^2\sigma^2 N}} + \sum_{t=t_1}^{\infty} e^{-N[\theta_1 b + t\epsilon]} \\
&\leq (t_1 - 1)\frac{e^{-N\min_{t<t_1}\sup_\theta[\theta b - \theta t[\alpha_1(\theta,t) - \frac{1}{N}\beta(\theta,t)]}}{\sqrt{2\pi\theta^2\sigma^2 N}} + \frac{e^{-N[\theta_1 b + t_1\epsilon]}}{1 - e^{-N\epsilon}}
\end{aligned}$$

Since the second term of the expression above is a fixed value and for large N we get,

$$P\left\{W^{(N)} > Nb\right\} \approx \frac{e^{-NI(b)}}{\sqrt{2\pi\theta^2\sigma^2 N}} \tag{31}$$

The term $\sigma^2\theta^2$ can be approximated by $2I(b)$ [15]. thereby resulting in the following approximation from eq.(31):

$$P\left\{W^{(N)} > Nb\right\} \approx \frac{e^{-NI(b)}}{\sqrt{4\pi NI(b)}} \tag{32}$$

Hence yielding the desired result.

Proof of Lemma 4

$$\begin{aligned}
\beta^{(S^1\otimes S^2)(t)}(\theta,t) &= -\frac{1}{\theta t} log E\left[e^{-\theta(S^1\otimes S^2)(t)}\right] \\
&= -\frac{1}{\theta t} log(\sum_{z=-\infty}^{\infty} e^{\theta z} P(\sup_{0\le s\le t}[-S^1(t-s)-S^2(s)] = z)) \\
&\ge -\frac{1}{\theta t} log(\sum_{z=-\infty}^{\infty}\sum_{s=0}^{t} e^{\theta z} P([-S^1(t-s)-S^2(s)] = z)) \\
&\ge -\frac{1}{\theta t} log((t+1)\sup_{0\le s\le t} E\left[e^{-\theta[S^1(t-s)+S^2(s)]}\right]) \\
&= -\frac{log(t+1)}{\theta t} + \inf_{0\le s\le t}[\frac{(t-s)}{t}\beta^1(\theta,t-s) + \frac{(s)}{t}\beta^2(\theta,s)] \\
&= -\frac{log(t+1)}{\theta t} + \frac{1}{t}(t\beta^1(\theta)\otimes t\beta^2(\theta))(t)
\end{aligned}$$

proves the claim.

Proof of Lemma 5

$$\begin{aligned}
\alpha^{(A\oslash S)(t)}(\theta,t) &= \frac{1}{\theta t} log E\left[e^{\theta(A\oslash S)(t)}\right] \\
&= \frac{1}{\theta t} log(\sum_{z=-\infty}^{\infty} e^{\theta z} P(\sup_{0\le s}[A(t+s)-S(s)] = z)) \\
&\le \frac{1}{\theta t} log(\sum_{z=-\infty}^{\infty}\sum_{s=0}^{\infty} e^{\theta z} P([A(t+s)-S(s)] = z)) \\
&\le \frac{1}{\theta t} log(\sum_{s=0}^{\infty} e^{\theta[(t+s)\alpha(\theta)-s\beta(\theta)]}) \\
&= \frac{1}{\theta t} log(\frac{e^{\theta t\alpha(\theta)}}{1-e^{\theta(\alpha(\theta)-\beta(\theta))}}) \\
&= \alpha(\theta) - \frac{1}{\theta t} log[1-e^{\theta(\alpha(\theta)-\beta(\theta))}]
\end{aligned}$$

proves the claim.

Worst Case Analysis for Multiple Priorities in Bitwise Arbitration

Ulrich Klehmet, Thomas Herpel, Kai-Steffen Hielscher, Reinhard German
Department of Computer Science 7
(Computer Networks and Communication Systems)
University of Erlangen-Nürnberg
Martensstraße 3
D-91058 Erlangen, Germany
Email: {klehmet, herpel, ksjh, german}@informatik.uni-erlangen.de

Abstract

As a widespread automotive network with bitwise arbitration, CAN buses are deployed in modern cars to fulfill the demands of more than 90 collaborating electronic control devices. For safety critical applications, fast and reliable data transfer is indispensable, since often hard real-time transmission deadlines have to be met to assure a safe operation of the vehicle. Therefore deterministic performance evaluation methods are inevitable for the validation of systems that must guarantee hard delay bounds and timeliness of information processing. One more recent deterministic modeling approach is Network Calculus, which allows to determine worst case transmission times. Based on real-world communication CAN bus data we could generate appropriate modeling elements for the Network Calculus as arrival- and service curves that reflect all priorities of CAN traffic. Until now, it was just known that the message with highest priority on a CAN bus has real-time properties. Now, the results of this paper allow for calculating delay bounds of the messages on all priority levels.

1. Introduction

Modern cars are equipped with various electronic control units (ECUs), enabling on-board entertainment and infotainment, wireless connectivity or enhancing the ability to actively avoid crashes or at least mitigate the consequences as far as possible.
Besides the increasing power consumption, the communication and data exchange between the single control units is one of the most critical tasks to be considered, when expanding the electronic functionality inside the car.

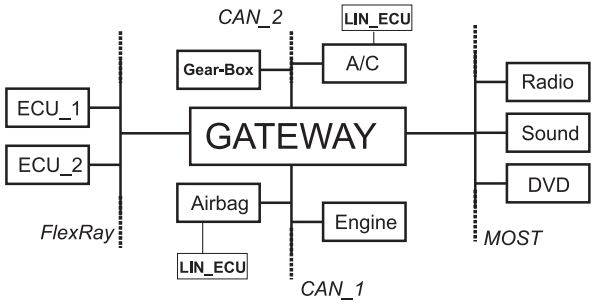


Figure 1. Intra-car communication network

Especially safety-critical functions, e.g. ESP (Electronic Stability Program), ABS (Antilock Braking System) or ACC (Adaptive Cruise Control), demand reliable, robust and high-speed performing data transmission in order to work efficiently. With the properties as described in detail in the following section, the CAN (Controller Area Network) bus [1] offers a sustainable performance to fulfill these demands, especially in terms of available data rate and collision avoidance at media access.
Thus, the majority of European carmakers use CAN based data buses and for Asian and American manufacturers, it is becoming more and more attractive, too. Besides CAN there are several other competing bus systems, often coexisting with CAN inside a car, e.g. LIN (Local Interconnect Network) [2], MOST (Media Oriented Systems Transport) [3] or FlexRay [4].
They differ in terms of data rate, media access and multiplexing schemes and other, mostly hardware related, peculiarities. Figure 1 shows a typical intra-car communication network as a mixture of different bus types connected by a central gateway. Here, the gateway, often installed in the dashboard, is the central network node, routing data frames between the single de-

vices and sub-bus systems. Other configurations with two or more gateways are possible, leading to a hierarchical structure with often autonomous sub-systems.
One problem of current network installations as in figure 1 is, that as the system expands, each additional electronic device consumes communication resources. Adding up the overall data traffic in a current midclass limousine, certain sub-bus systems are working at a high level bus load to serve the extended functionality.
Hence, investigating data traffic with respect to bus specific parameters, network topology and communication paradigms is a necessary task in twofold sense. On the one hand, the results of the investigations can help keeping the communication reliable and robust, especially for safety-relevant data, by adjusting the traffic settings appropriately. On the other hand, introducing new electronic features will be an ongoing process which can be performed successfully only if the existing communication infrastructure is able to bear the increasing data traffic load.
The objective of the investigations within this paper is to show the basic applicability of the method of Network Calculus for evaluation of automotive communication systems. In such systems the mean values for performance measures, as obtained from stochastic modeling approaches like Queuing Theory are of minor interest, as they are not sufficient to predict whether hard real-time deadlines are matched in any case. Reliable operation and avoidance of system-malfunctions with catastrophic effects mainly depend on the worst case performance of the communication infrastructure. The application of a deterministic modeling technique like Network Calculus yields upper delay bounds for data transmission, which are inevitable to assess the reliability of the system in all possible scenarios of operation, since the *timeliness* itself is an important aspect of hard real-time systems.

The paper is organized as follows: In section 2, the main features of the CAN bus, especially regarding the media access scheme, are presented briefly. Section 3 introduces the basic idea of Network Calculus, how actual input data can be used to construct the modeling elements arrival and service curve and how to determine delay bounds from these. Section 4 applies the findings from the previous section to both a short exemplary data set and the real-world CAN bus traffic data as obtained from Audi. Finally, section 5 summarizes the results and gives an outlook for future work on this topic.

2. The CAN Bus

CAN development has been started in the 1980s by Robert Bosch GmbH aiming to design a bus system for the specific needs of automotive applications. The resulting CAN bus is now standardized as ISO 11898 [1].

It uses a differential serial line architecture with *dominant* and *recessive* bits where a dominant bit represents a logical 0 and a recessive bit a logical 1. An idle bus has recessive level. Due to the open collector logic, if one station on the bus sends a dominant bit while another controller sends a recessive bit, the dominant bit wins, i.e. the bus is considered as logical 0. This feature allows to use a bit-wise arbitration scheme for the medium access, often called *CSMA/BA* (Carrier Sense Multiple Access with Bitwise Arbitration). Using this mechanism, each station listens to the bus while sending data. If a collision occurs where one station tries to send a recessive bit but receives a dominant bit, it will notice that another station is sending simultaneously and will immediately stop its own transmission. This makes the arbitration non-destructive, since the station sending the dominant bit can continue to send without any negative effects on the bus while the station sending the recessive bit remains silent from the time on where the collision has occurred. A retransmission of the interrupted frame is automatically triggered. Before sending to the bus, each controller listens to the bus and starts sending only if the bus has been recessive for at least 6 bit times (CS phase). A *NRZ* (Non-Return-to-Zero) encoding is used for the line encoding of the bits, where the sender inserts a complementary stuff bit when no change in the logic level has occurred over five successive bits. These stuff bits are automatically removed by the receiver of the message. This mechanism provides a base for synchronization and assures that a potential sender receives at least one dominant bit during the six bit times carrier sense phase if another station is sending.

CAN does not employ explicit sender or receiver addresses, but uses unique *message identifiers* to describe the content of the respective CAN frame. The frames are broadcast on the bus and each station can decide if the message content is relevant by examining the message identifier of received frames. These identifiers have to be assigned statically during the design phase of the bus system to avoid ambiguity in the interpretation of the frame content. A global view of the complete system is needed in this process.

There are two variants of CAN frames: standard frames with 11 bit message identifiers and extended frames with 29 bit identifiers. Both can coexist on the same bus.

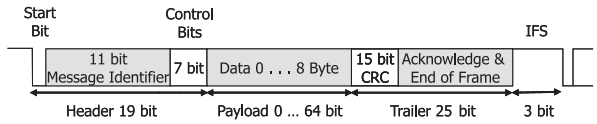


Figure 2. CAN Frame Structure

The payload can be of variable size, but for our application, all frames are standard CAN frames with 11 bit message identifiers and always contain 64 bits of payload. Thus, when considering a maximum number of 19 stuff bits and including 3 bit times of inter-frame space (IFS), the maximum frame size is 130 bits.

The transmission of a CAN frame starts with one dominant start bit immediately followed by the message identifier from the most significant to the least significant bit. The structure of a standard CAN frame is shown in figure 2. Due to the media access scheme described above, the message identifiers create an implicit hierarchy of priorities. If more senders start to send simultaneously, the sender transmitting the frame with the highest message identifier has to send a recessive 1 bit before any other sender due to the binary encoding of the message identifiers in the frame header. While listening to the bus during the send process, it will notice the collision (dominant bus while sending a recessive bit) and stop sending. This process will continue until only one sender remains sending. This is always the one with the lowest message identifier and thus with the highest priority. [5, 6]

3. Network Calculus

Network Calculus is a system theory for deterministic queuing systems, based on min-plus algebra [7],[8] and [9]. It plays a role similar to classical system theory for the analysis of electronic circuits where in Network Calculus addition is replaced by computation of the infimum and multiplication becomes addition. Its main focus is on determination of bounds on worst case performance. The aim for example is to get lower and upper bounds for end-to-end delays of nodes or collection of nodes within a network, for traffic backlog and for output limitations. By means of these performance-analytic bounding values – characterizing worst-case behavior of traffic flows – it is possible to dimension the corresponding buffers e.g. or to limit bursts etc. In the sequel we will give only a short introduction into Network Calculus. A comprehensive overview on it can be found in [7]. The most important modeling elements are the *arrival* and the *service curves* together with the *min-plus convolution*.

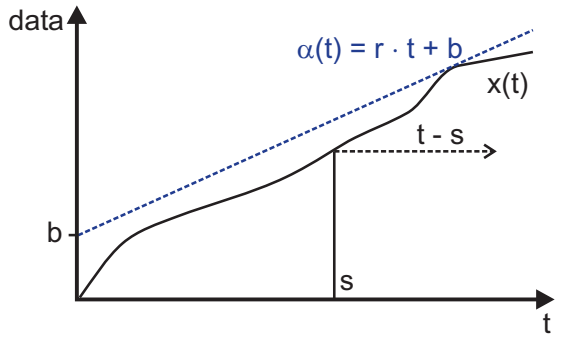


Figure 3. Token Bucket Arrival Curve

Let F be a flow (bits, messages, packets, etc.) into a system S and let $x(t)$ be the amount of data of F arriving in time interval $[0, t]$ and $y(t)$ the amount of data leaving S in time interval $[0, t]$. $x(t)$ is the arrival function of flow F. By definition $x(0)$ is zero and $x(t) \geq x(s)$ for all $t \geq s$. The backlog and delay for a lossless system are described as follows: The *backlog at time t* is the amount of data just in transit between input and output of a system, given by $x(t) - y(t)$, where the (virtual) *delay at time t* is $d(t) = \inf\{\tau : x(t) \leq y(t + \tau)\}$.

An upper bound on the arrival function $x(t)$ can be defined by the so-called arrival curve $\alpha(t)$:

Definition 1 (Arrival curve)**.** Let $\alpha(t)$ be a non-negative, non-decreasing function. Flow F is constrained by or has arrival curve $\alpha(t)$ iff $x(t) - x(s) \leq \alpha(t - s)$ for all $t \geq s \geq 0$.

Example. A commonly used arrival curve is the token bucket constraint:

$\alpha_{r,b}(t) = b + rt$ for $t > 0$ and zero otherwise.

As one can see in figure 3 this arrival curve forms an upper limit for traffic flows $x(t)$ with (average) rate r and instantaneous burst b . That means $x(t) - x(s) \leq \alpha_{r,b}(t - s) = b + r(t - s)$. For $\Delta t := t - s$ and $\Delta t \to 0$ it holds

$$\lim_{t \to s}\{x(t) - x(s)\} \leq \lim_{\Delta t \to 0}\{r \cdot \Delta t + b\} = b$$

An important definition in Network Calculus is the following one:

Definition 2 (Min-plus convolution)**.** Let $f(t)$ and $g(t)$ non-negative, non-decreasing functions that are 0 for $t \leq 0$. A third function, called min-plus convolution is defined by

$$(f \otimes g)(t) = \inf_{0 \leq s \leq t}\{f(s) + g(t - s)\}$$

Applying Definition 2 we can characterize the arrival curve $\alpha(t)$ with respect to $x(t)$ as:

$$x(t) \leq (x \otimes \alpha)(t)$$

The concept of arrival curves describes an upper bound to an input stream of a system processing some type of data. Concerning the output of this system we are interested in some service guarantees, i.e. is there a guaranteed minimum of output $y(t)$ – the amount of data leaving system S? The modeling element *service curve* deals with this problem.

Definition 3 (Service curve)**.** Given is a system S with input flow $x(t)$ and output flow $y(t)$. The system offers a (minimum) service curve $\beta(t)$ to the flow iff $\beta(t)$ is a non-negative, non-decreasing function with $\beta(0) = 0$ and $y(t)$ is lower bounded by the convolution of $x(t)$ and $\beta(t)$:

$$y(t) \geq (x \otimes \beta)(t).$$

Figure 4 demonstrates $(x \otimes \beta)(t)$ as an example for the lower bound of the output $y(t)$ and any given input $x(t)$.

Example. One commonly used service curve is the rate-latency function: $\beta(t) = \beta_{R,T}(t) = R \cdot [t - T]^+ := R \cdot \max\{0; t - T\}$. The rate-latency function reflects a service element which offers a minimum service of rate R after a worst-case latency of T. Having in mind a worst case performance analysis, it is possible to abstract away from complex (queuing) systems with different scheduling strategies.

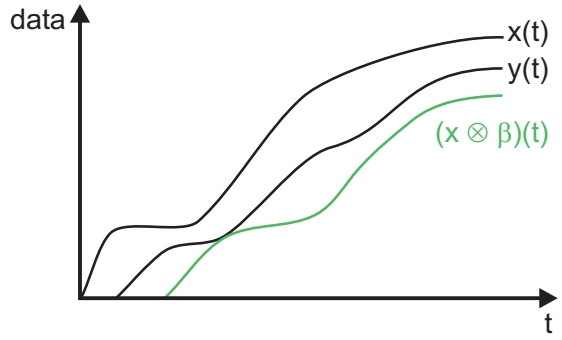


Figure 4. Convolution as Lower Output Bound

In figure 6, the graph $\beta_{R,T}(t)$ reflects a rate-latency service curve with rate R and latency T.

Theorem 1 (Backlog bound)**.** *Consider a system with input flow $x(t)$ and output flow $y(t)$. Let $x(t)$ be constrained to arrival curve $\alpha(t)$. Let the system offer a (minimum) service curve $\beta(t)$. The backlog v at time t, $v(t) = x(t) - y(t)$, is bounded by the supremum of the vertical deviation of arrival curve and service curve:*

$$x(t) - y(t) \leq \sup_{s \geq 0}\{\alpha(s) - \beta(s)\}.$$

The next theorem gives a bound on delay for the general case.

Theorem 2 (Delay bound)**.** *Assume a flow constrained by arrival curve $\alpha(t)$ passing a system with service curve $\beta(t)$. The maximal virtual delay d is given as the supremum of all possible virtual delays of data, i.e. is defined as the supremum of the horizontal deviation between arrival curve and service curve:*

$$d \leq \sup_{s \geq 0}\{\inf\{\tau : \alpha(s) \leq \beta(s + \tau)\}\}.$$

Figure 5 depicts both theorems.

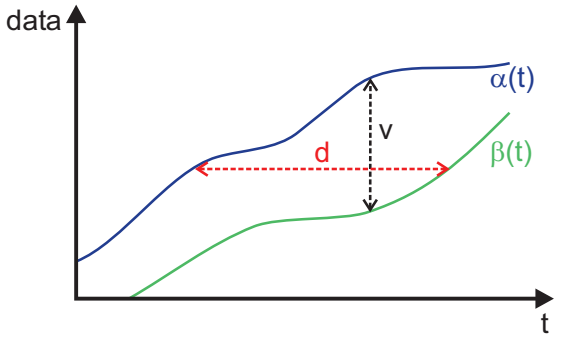


Figure 5. Backlog and delay bound

Example. Suppose there is a system with input according to token bucket, thus $x(t) - x(s) \leq \alpha_{r,b}(t - s)$ and rate-latency output:

$$y(t) \geq \inf_{s \leq t}\{x(s) + \beta_{R,T}(t - s)\}$$

Based on the above theorems we get for the delay bound $d \leq b/R + T$, and the backlog is bounded by $v = b + rT$. Figure 6 shows the results.

3.1. Input Data Basis

For a realistic system evaluation with Network Calculus, it is necessary to use actual data. The input data are obtained from the *Department for Safety Electronics* at AUDI.
The object of study is the drivetrain CAN bus, as installed in up-to-date and future AUDI vehicles, working with a data rate of 500 kbps, which is typical for high-speed CAN communication. Important ECUs – engine, airbag and others – are directly connected to this sub-bus system.
To determine delay bounds, communication specific

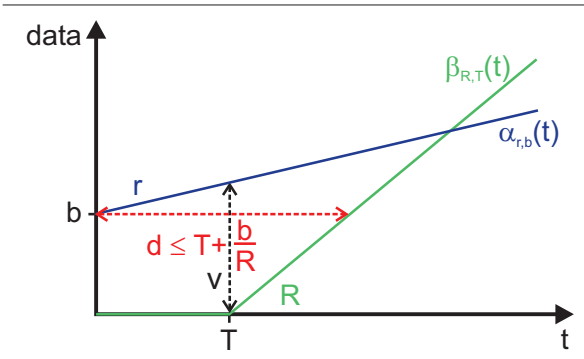


Figure 6. Example for the bounds

data are available, organized in a so-called Communication Matrix. The data set reflects the communication dependencies between control units connected to the drivetrain CAN and to other bus segments via the gateway. The traffic is comparted into the single CAN signals within the CAN messages, each row of the matrix corresponds to a certain CAN signal, whereas the columns contain signal-dependent bus parameters such as cycle time, CAN priority and transmission type particularities (cyclic, dynamic, onChange, opening). In addition, physical parameters for all signals of a message are listed, e.g. the length in bit, defined signal values and corresponding bit patterns or offsets, units, scalings and raw values.

From the matrix, an overall number of 62 active CAN messages can be figured out, restricted to messages which are sent cyclic when the start-up phase of the car is over. This restriction is valid, since messages that are sent acyclic only serve for establishing bus communication after engaging of the ignition switch or for purposes of on-board diagnostics at repair-shops. They do not contribute to CAN bus traffic load during runtime in road traffic, for which Network Calculus is used to estimate delay bounds of in-car communication. Since each message is provided with a unique CAN identifier, the 62 different CAN priority classes can easily be derived by ordering the IDs from lowest (highest priority) to highest (lowest priority).
As shown later, information about the priority class and the bus cycle time of the single messages is sufficient to apply Network Calculus for performance evaluation of the drivetrain CAN.

Determining reasonable bounds for delays of safety-critical data transmission depends not only on the performance and access mechanism of the communication system, but also requires reliable data transfer with respect to transmission errors and retransmissions, robustness of the bus system and control units, e.g. in terms of electromagnetic compatibility, and appropriate concepts for safe and efficient processing of the transmitted data.

3.2. Generation of Arrival Curves

A central issue of performance evaluation with Network Calculus is the generation of appropriate arrival curves for data traffic. As explained before, the media access in CAN communication is performed in a cyclic manner. Successful transmission depends on the priority of the message to be sent compared to the priority of data, which is transfered via the bus simultaneously. Thus, from the point of view of a certain message with priority n, two types of arrival curves have to be distinguished.
The higher priority classes from 0 to $n-1$ form an own arrival curve. It represents the cumulative arrivals of messages along a specified time interval, during which no media access for class n is possible due to higher prioritized traffic. For lower priorities, only arrivals of class n must be considered, since all even lower priorities ($n+1$ to lowest) are dominated by messages of class n. The highest priority 0 has to be treated separately, since here no arrival curve for higher classes exists and hence the arrival curve for higher classes is constantly zero.
To clarify the idea, the generation of arrival curves for the 62 drivetrain CAN priorities is explained in detail in the following. The generation of arrival curves is based on the assumption that at each discrete point of time, all messages with matched cycle period occur at once. Hence, the lower the priority of the message, the more higher prioritized data may access the bus at the same time, prolongating the time until media access will be successful.
This approach is valid, since performance evaluation with Network Calculus aims at determination of upper (worst-case) delay bounds. Of course, scheduling the data access during start-up in a reasonable way might yield reduced delays. This could be done by a proper scheduling.

Let N denote the highest priority used and $n \in \{0, 1, \ldots, N\}$ be a message identifier and thus a priority for one particular CAN frame. Then the data provided by AUDI shows that this message will be sent in cycles of length T_{c_n}. A new message of priority n will be sent immediately after times $k \cdot T_{c_n}, k \in \mathbb{N}_0$, i.e. at integer multiples of T_{c_n}. Let T_C denote the least common multiple (lcm) of all T_{c_n}. At this point of time, messages of all priorities are sent, and since all messages are sent in a cyclic manner, the cycle of mes-

sage arrivals will repeat in the same sequence for all messages of all priorities after the total cycle time $T_C = \text{lcm}\{T_{c_0}, T_{c_1}, \ldots T_{c_N}\}$.

The arrival curve for message n can be written as the following *step function*:

$$\alpha_n(t) = \left\lceil \frac{t}{T_{c_n}} \right\rceil \cdot l,$$

where $l = 136$ bits denotes the maximum frame length including the 6 bit CS time.

The cumulative number of arrivals of messages with priority higher than n can be calculated as

$$\begin{aligned} \hat{\alpha}_n(t) &= \sum_{i=0}^{n-1} \alpha_i(t) \\ &= \sum_{i=0}^{n-1} \left\lceil \frac{t}{T_{c_i}} \right\rceil \cdot l. \end{aligned}$$

A linear function $\bar{\alpha}_n(t)$ can be determined as an upper bound for the cumulative higher priority arrivals $\hat{\alpha}_n(t)$ as

$$\begin{aligned} \bar{\alpha}_n(t) &= n \cdot l + \frac{\sum_{i=0}^{n-1} T_C/T_{c_i} \cdot l}{T_C} \cdot t \\ &= n \cdot l + \sum_{i=0}^{n-1} \frac{l}{T_{c_i}} \cdot t \\ &= \sum_{i=0}^{n-1} \left(\frac{t}{T_{c_i}} + 1 \right) \cdot l \end{aligned}$$

We construct the curve of $\bar{\alpha}_n(t)$ using the following idea: We define a token bucket arrival curve $\bar{\alpha}_n(t)$. Therefore, we have to come up with the vertical offset and the slope of the curve. The offset is obtained by adding all single offsets (i.e., we assume that all possible bursts happen initially immediately after time $t = 0$). The slope is obtained by adding the slopes bounding all single higher priority arrival curves. These slope boundaries are given by $\frac{T_C/T_{c_i}}{T_C} \cdot l = l/T_{c_i}$, because exactly T_C/T_{c_i} messages of length l with priority i are generated in each interval of length T_C. As one can easily see, $\bar{\alpha}_n(t) \geq \hat{\alpha}_n(t)$. Thus, $\bar{\alpha}_n(t)$ is an upper bound for the cumulative arrival of higher priority messages that fulfills all preconditions for a Network Calculus arrival curve. Defining $b_n := n \cdot l$ and $r_n := \sum_{i=0}^{n-1}(l/T_{c_i})$ allows us to write this function in token bucket form

$$\bar{\alpha}_n(t) = b_n + r_n \cdot t.$$

The second column of figure 8, where the step function $\hat{\alpha}_i(t)$ is drawn as a solid line and the linear function $\bar{\alpha}_i(t)$ as a dashed line, illustrates both cumulative curves for different priorities.

3.3. Determination of Service Curves and Calculation of Delay Bounds

Now it will be shown how the Network Calculus can be applied to compute the time important maximum frame transfer delay of one bus controller station. After we determined a step function as an arrival curve for the input traffic in section 3.2, we have to look for a 'good' service curve reflecting the real condition of a CAN node as far as possible. Is it realistic to find a rate-latency service curve $\beta_{R,T}(t)$ for the outgoing data like in many other worst-case modeling situations? Yes, we think so.

First of all, for a high-speed CAN the rate R is given by 500 kbps for all message frames, independent of their priority. Considering the inherent hierarchy of frame priorities and the non-preemptive scheduling mechanism of data sending the following holds: Assuming the worst case, a higher priority frame always has to wait until a frame, possibly of lower priority, is completely sent. Since the maximum frame size is 130 bits and the CS time is 6 bit times, the worst-case sending time of such lower priority data is $T = l/R = 136$ bits/500 kbps $= 0.272$ ms.

Thus, for the rate-latency service curve $\beta_{R,T}(t)$ it holds:

$$\begin{aligned} \beta_{R,T}(t) &= R \cdot [t - T]^+ \\ &= 500 \text{ kbps} \cdot [t - 0.000272 \text{ s}]^+. \end{aligned}$$

The next problem we have to solve deals with the priority scheduling of data frames. Because there are 11 bits in the message identifier (cf. figure 2) for standard CAN frames, 2^{11} different priorities are possible at most. The highest priority is the one with number 0 and the lowest is $2^{11} - 1$. Based on theorem 2 computing the guaranteed worst case delays of all these priority-different traffic, we use the arrival and service curve only. Here, we propose the following approach:

Let us denote the input or output data at time t per controller by $x_j(t)$ and $y_j(t)$ respectively, with priority $j \in \{0, \cdots, 2^{11} - 1\}$. The procedure starts with input $x_0(t)$ and output $y_0(t)$. Theorem 2 tells us that delay

$$D_0 \leq \sup_{t \geq 0}\{\inf\{\tau : \alpha_0(t) \leq \beta(t + \tau)\}\}$$

with $\alpha_0(t)$ = step-function-type arrival curve of input $x_0(t)$ and $\beta(t) = \beta_{R,T}(t) = 500$ kbps$\cdot[t - 0.000272 \text{ s}]^+$. Then we have to determine the maximum delay D_1 for the next lower priority frames of the same controller station. However, it would not be correct to take the same service curve $\beta(t)$, because the frames of priority 1 (pr1-frames) will be served only after sending of pr0-frames is finished, cf. figure 7. We follow the approach

of aggregate traffic modeling in [10] and construct the service curve

$$\beta_1(t) := [\beta(t) - \bar{\alpha}_1(t)]^+ .$$

Even though $\hat{\alpha}_1(t) = \alpha_0(t)$ is a valid arrival curve, we need to use $\bar{\alpha}_1(t)$ instead of $\hat{\alpha}_1(t)$ to guarantee that $\beta_1(t)$ is a non-decreasing function, since $\bar{\alpha}_1(t)$ is a linear token bucket type function.

The maximum delay D_1 of pr1-frames is

$$D_1 \leq \sup_{t \geq 0} \{\inf \{\tau : \alpha_1(t) \leq \beta(t+\tau) - \bar{\alpha}_1(t+\tau)\}\} .$$

Going on in this way from priority j to the next lower one $j+1$ we must build a new service curve by diminishing the previous one by the possible maximum of all higher-priority frames, i.e. by the cumulative arrival curve $\bar{\alpha}_{j+1}(t)$ for frames with priority $0, 1, \ldots, j$. So, at each priority step only two sorts of frames are considered: the frames of present priority j and the sum of all higher priority frames – all frames of lower priority $(j+1, \cdots, 2^{11})$ have no impact besides maybe a single frame just in service which has already been modeled by the 'non-preemptive' scheduling in the latency component T of $\beta_{R,T}(t)$.

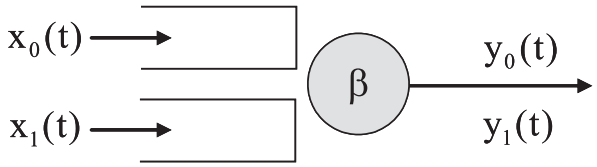


Figure 7. Non-preemptive Priority Net Element

The following procedure summarizes our approach: $\forall j \in \{0, 1, \ldots, 2^{11} - 1\}$ and $t \in [0, \infty]$:

$$\begin{aligned}
\alpha_j(t) &= \left\lceil \frac{t}{T_{C_j}} \right\rceil \cdot l \\
\bar{\alpha}_j(t) &= j \cdot l + \sum_{i=0}^{j-1} \frac{l}{T_{c_i}} \cdot t \\
&= b_j + r_j \cdot t \\
\beta_j(t) &= [\beta(t) - \bar{\alpha}_j(t)]^+ \\
&= \left[R[t-T]^+ - (b_j + r_j \cdot t)\right]^+ \\
&= [R(t-T) - (b_j + r_j \cdot t)]^+ \\
&= \left[(R - r_j) \cdot \left(t - \frac{RT + b_j}{R - r_j}\right)\right]^+
\end{aligned}$$

Again, we get a rate latency service curve with rate $R'_j := R - r_j$ and latency $T'_j := (RT + b_j)/(R - r_j)$.

Lastly, we get

$$D_j \leq \sup_{t \geq 0} \{\inf \{\tau \geq 0 : \alpha_j(t) \leq \beta_j(t+\tau)\}\}$$

Due to the actual shape of the arrival curve $\alpha_j(t)$ and the service curve $\beta_j(t)$ in our case, this inequation can be solved geometrically by calculating the intersection point of $\beta_j(t)$ with the height of the first step of the arrival curve $\alpha_j(t)$, which is exactly $l = 136$ bits. This yields the following results:

$$\begin{aligned}
D_j &\leq \frac{l}{R'_j} + T'_j \\
&= \frac{l + RT + b_j}{R - r_j} \\
&= \frac{(j+2) \cdot l}{R - \sum_{i=0}^{j-1} (l/T_{c_i})}
\end{aligned}$$

4. Application

4.1. Exemplary Message Schedule

To illustrate the process of generating the arrival curves and calculating the service curves and delays, assume that only five different CAN priorities with respective cycle times as shown in table 1 are used and that all messages are standard CAN frames with a maximum length of 130 bits, therefore, $l = 136$ bits.

Priority i	Cycle Time T_{c_i}
0	50 ms
1	10 ms
2	100 ms
3	20 ms
4	30 ms

Table 1. Example Priorities and Cycle Times

Table 2 shows the results obtained for a CAN data rate of 500 kbps.

Figure 8 illustrates these results. The arrival curve $\alpha_i(t)$ for each priority i is shown in the first column of this figure. In the second column, the step function $\hat{\alpha}_i(t)$ for the cumulative number of higher priority arrivals is drawn as a solid line and the higher priority arrival curve $\bar{\alpha}_i(t)$ as a dashed line. The third column shows the arrival curve $\alpha_i(t)$ as a solid line and the corresponding service curve $\beta_i(t)$ as a dashed line. The horizontal distance between the vertical dotted lines marks the geometrically determined maximum delay D_i as the intersection point of $\beta_i(t)$ and $l = 136$ bits.

		$\bar{\alpha}_i(t)$		$\beta_i(t)$		
i	T_{c_i} [ms]	b_i [bits]	r_i [bps]	R'_i [bps]	T'_i [ms]	D_i [ms]
0	50	0	0	500000	0.272	0.544
1	10	136	2720	497280	0.547	0.820
2	100	272	16320	483680	0.844	1.125
3	20	408	17680	482320	1.128	1.410
4	30	544	24480	475520	1.430	1.716

Table 2. Example Results

Please note the different scale of both the time and the data axis in the third column compared to the first two columns in this figure. The intersection point would not be visible due to the different orders of magnitude in temporal and data dimension of the arrival curve and the service curve otherwise.

4.2. Actual CAN Traffic

Figure 9 illustrates the computed maximum delays for all CAN frames in the AUDI data set. As one could expect, the lower the priority (increasing priority number) the larger is the maximum possible delay.

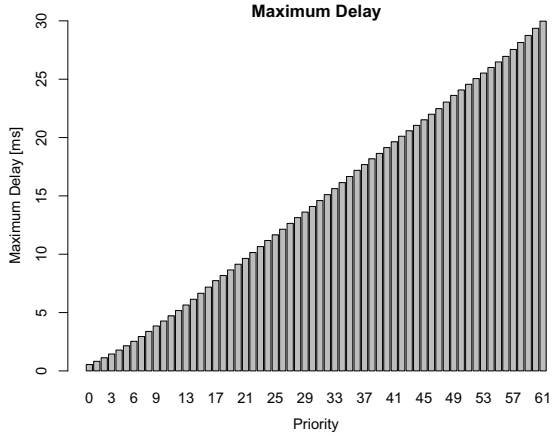


Figure 9. Maximum Delays for All Priorities

5. Conclusions and Future Work

We have shown in this paper, that applying the method of Network Calculus yields reasonable results for deterministic performance evaluation of state-of-the-art automotive communication networks. The delay bounds for CAN messages we obtained support the decision, whether for arbitrary communication dependencies the hard real-time operation bounds can be matched in any constellation. Our approach marks a noticeable progress in investigations of in-car communication, as it provides an analytical method to determine delay bounds for all CAN priority classes. Up to now, this was only feasible for messages from the highest CAN priority level, which are already known to have real-time transmission properties.

Thus, for future system layouts more sophisticated network evaluation with respect to dependability and functionality, especially for safety-critical applications, is possible. Further research on this topic will concentrate on a refinement in modeling of arrival- and service-curves. The challenge is to construct upper-bound thresholds, that resemble the actual data traffic even closer, while they still fulfill all conditions to be modeling elements of Network Calculus. As a consequence, we expect to obtain even more precise delay limits as from the first-step evaluation approach within this paper.

References

[1] International Standard ISO 11898, *Road vehicles – Interchange of Digital Information – Controller Area Network (CAN) for High Speed Communication*, 1st ed. International Organization for Standardization, 1994, ISO Reference Number ISO 11898:1993(E).

[2] H.-C. v. d. Wense, Ed., *LIN Specification Package*. LIN Consortium, 2003.

[3] MOST Cooperation, *MOST Media Oriented Systems Transport*, rev 2.4 ed., 2005.

[4] FlexRay Consortium, *FlexRay Communications System Protocol Specification*, version 2.1 ed., 2005.

[5] W. Lawrenz, Ed., *CAN Controller Area Network*, 4th ed. Heidelberg: Hüthig Verlag, 2000.

[6] K. Etschberger, *Controller-Area-Network*, 2nd ed. München: Carl Hanser Verlag, 2000.

[7] J.-Y. Le Boudec and P. Thiran, *Network Calculus*. Springer Verlag LNCS 2050, 2001.

[8] R. Cruz, "A calculus for network delay, part i: Network elements in isolation," *IEEE Trans. Inform. Theory*, vol. 37-1, pp. 114–131, 1991.

[9] ——, "A calculus for network delay, part ii: Network analysis," *IEEE Trans. Inform. Theory*, vol. 37-1, pp. 132–141, 1991.

[10] M. Fidler and V. Sander, "A parameter based admission control for differentiated services networks," *Computer Networks*, vol. 44, pp. 463–479, 2004.

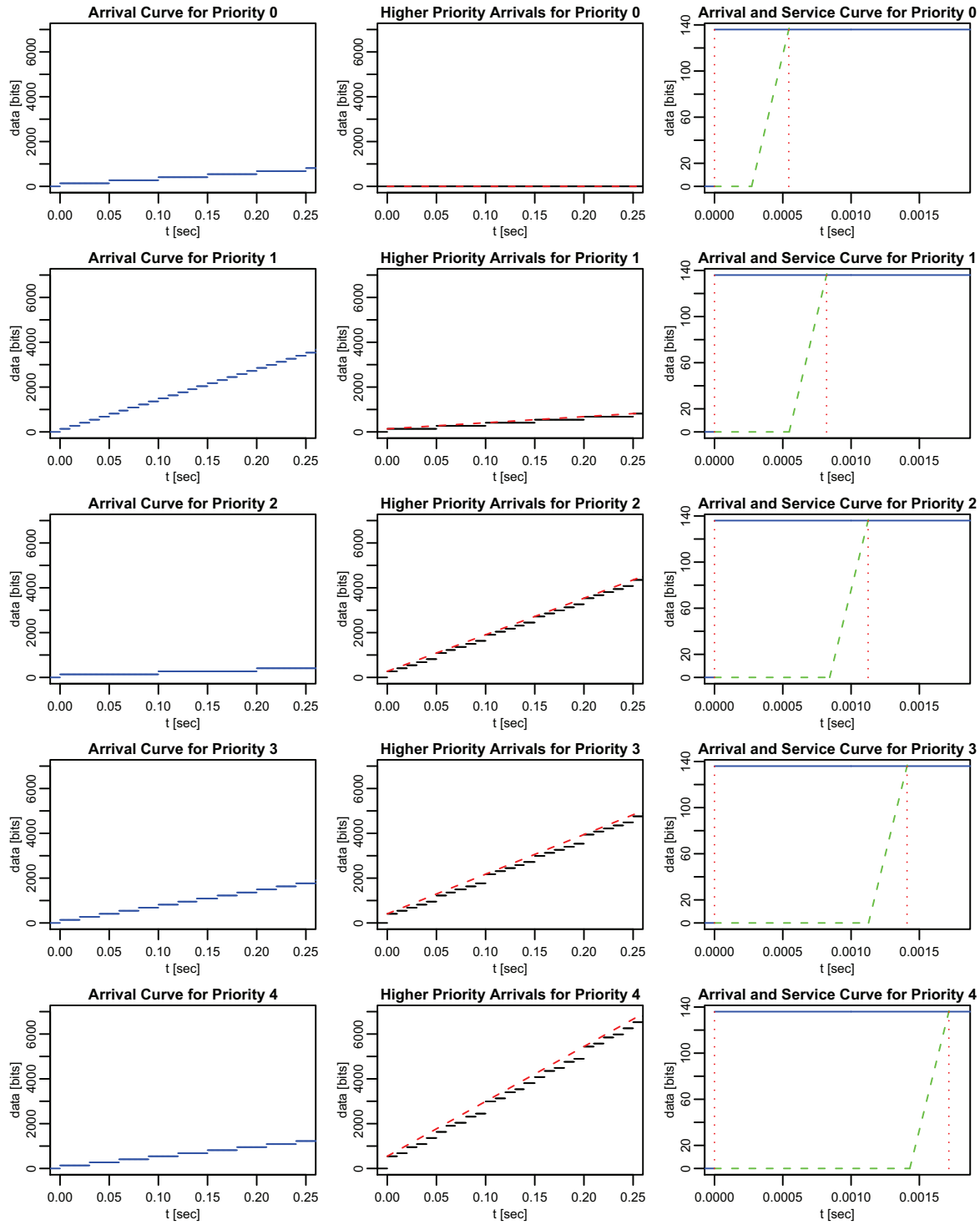


Figure 8. Analysis Results for the Example

Modellierung verlustinduzierender Lasttransformationen für markovsche Ankunftsprozesse

Stephan Heckmüller, Bernd E. Wolfinger

Dept. Informatik, TKRN, Universität Hamburg
Vogt-Kölln-Str. 30, D-22527 Hamburg
E-mail: {heckmueller|wolfinger}@informatik.uni-hamburg.de

Kurzfassung

Realistische Lastmodellierung ist für eine exakte Leistungsbewertung von Kommunikationsnetzen unverzichtbar. Innerhalb dieser Netze erfolgt die Diensterbringung typischerweise durch eine Reihe von Zwischenschritten, welche die Lasteigenschaften, wie sie von nachfolgenden Stationen gesehen werden, beinflussen. In diesem Beitrag werden diese Zwischenschritte unter Verwendung des Konzepts der Lasttransformation explizit modelliert. Neben der Präsentation der allgemeinen Methodik erfolgt die Anwendung auf markovsche Ankunftsprozesse als Lastmodelle. Hierbei gilt die Konzentration einer Lasttransformation, welche insbesondere in drahtlosen Netzen von hervorragender Bedeutung ist, dem Auftragsverlust. Diese Auftragsverluste werden hierbei direkt in die Lastbeschreibung integriert, ohne die analytische Handhabbarkeit zu verlieren. Dies gilt sowohl für unabhängig verteilte Verluste als auch für markovsche Verlustprozesse. Darüber hinaus werden die Kombinationen verschiedener Transformationen und die hieraus resultierende Ankunftsrate untersucht.

1 Einleitung und Problemstellung

Netzwerklasten, wie sie von heutigen (verteilten) Anwendungen induziert werden, weisen im Allgemeinen komplexe statistische Eigenschaften auf, welche wiederum einen starken Einfluss auf die Leistungskennwerte realer Systeme haben. Neben der Entstehung neuer Leistungsbewertungsverfahren und Messtechniken sind daher auch im Bereich der Lastmodellierung eine Vielzahl von Neu- und Weiterentwicklungen zu verzeichnen.
Unglücklicherweise werden die Lasteigenschaften, wie sie an höheren Schnittstellen einer Netzarchitektur festgestellt werden können, durch die Verarbeitungsschritte innerhalb des übertragenden Rechnernetzes in signifikanter Weise verändert. Zu diesen Schritten zählen Fragmentierung, Header-Generierung und Maßnahmen zur Überlastvermeidung. Die Methode der Lasttransformation bietet die Möglichkeit diese Schritte explizit zu modellieren, um so zu einer Beschreibung der Lasteigenschaften an tieferen Schnittstellen des Netzes zu gelangen. Neben den genannten Verarbeitungsmechanismen kann eine Beeinflussung der Lasteigenschaften durch fehlerhafte Verarbeitung erfolgen, wobei im Kontext der Rechnernetze insbesondere Paketverluste zu nennen sind. Diese dienen im Gegensatz zu den vorgenannten Verfahren nicht der Diensterbringung, auch wenn das Verwerfen von Paketen zur Überlastvermeidung ein Mittel zur Aufrechterhaltung der Dienstbereitschaft eines Rechnernetzes ist.
Im vorliegenden Beitrag soll gezeigt werden, dass sich eine Reihe wichtiger Verlustprozesse – unabhängig verteilte Verluste und markovsche Verlustprozesse – direkt als Lasttransformationen auf markovschen Ankunftsprozessen angeben lassen. Somit kann die analytische Handhabbarkeit der transformierten Lastmodelle sichergestellt werden, bei gleichzeitiger Beibehaltung guter Realitätsnähe.
Das Konzept der Lasttransformation ist bereits in einer Reihe von Arbeiten untersucht worden [WZHB02, Bai99, Zad01, HW07]. Die in den vorgenannten Arbeiten verwendete Lastbeschreibungstechnik legt besondere Betonung auf die Schnittstelle, welche das Bediensystem (z.B. ein paketübertragendes Netz) von seiner Umgebung (z.B. die Netzdienste beanspruchenden Benutzer) separiert. Hierdurch wird die Lastbeschreibung in Abhängigkeit vom betrachteten Bediensystem möglich. Hierauf aufbauend führen die Autoren die Unterscheidung von Primär- und Sekundärlast ein, welche die formale Definition der Lasttransformation ermöglicht. In [WZHB02] werden analytische Lasttransformationen für Rechnernetze untersucht und auf ihre Realitätsnähe überprüft. Die Autoren konzentrieren sich auf die Transformation

von Paketlängen, wie sie durch Fragmentierung und Header-Generierung hervorgerufen wird. Weiterhin wird in [Bai99] die simulative Lasttransformation untersucht. In [HW07] werden Transformationen auf markovschen Ankunftsprozessen (BMAPs) für Fragmentierung und Schiebefenstermechanismen vorgeschlagen, welche sowohl die Transformation von Paketattributen als auch des zeitlichen Verhaltens modellieren.
Batch Markovian Arrival Processes (BMAP) wurden in abweichender Notation in [Neu79] eingeführt. Der Autor verwendet die Bezeichnung *versatile Markovian Point Process*. BMAPs in der in vorliegender Arbeit verwendeten Notation wurden in [Luc91] vorgeschlagen. Aufbauend auf diesen Arbeiten wurden BMAPs in einer Vielzahl weiterer Arbeiten untersucht. Betrachtet wurden beispielsweise die Blockierungswahrscheinlichkeit einer $BMAP/G/1$-Warteschlange [CW06], der Abgangsprozess einer $BMAP/G/1$-Warteschlange [FC01] oder die Wahrscheinlichkeitsverteilung der Arbeitslast von Systemen mit Schwellwert [LB06]. In letztgenannter Arbeit wird angenommen, dass der Bediener erst aktiv wird, wenn die Anzahl wartender Aufträge einen festzulegenden Schwellwert überschreitet. Eine Reihe anderer Arbeiten verallgemeinern die $BMAP/G/1$-Warteschlange: In [Hof01] wird die $BMAP/G/1$-Warteschlange erweitert, um insbesondere eine Abhängigkeit des Ankunftsstromes vom Systemzustand zu gestatten.
In [KLL03] werden BMAPs genutzt, um den Verkehr in Netzen auf IP-Ebene zu modellieren. Die Autoren schlagen hierzu Techniken zur Anpassung der Modellparameter an vorliegende Messdaten vor. Einige Arbeiten nutzen BMAPs zur Modellierung von Videolasten. In [BC92] werden Lasten mit variabler Bitrate (VBR) mit Hilfe von BMAPs modelliert. In [HSB04] werden Videolasten durch diskrete $BMAP$s modelliert, um ausgehend hiervon warteschlangen-basierte Untersuchungen vorzunehmen. Die so erzielten Ergebnisse dienen weiterhin der Dimensionierung von Puffern.
Die im Folgenden verwendeten markovschen Fehlermodelle finden insbesondere in nachrichtentechnischen Untersuchungen drahtloser Übertragungskanäle Verwendung. In [ZRM95] werden diese auf ihre Tauglichkeit zur Modellierung des 'Fadings' von Kanälen untersucht. Die Autoren sind der Ansicht, dass auch einfache Markov-Modelle als gute Approximationen des Kanalverhaltens angesehen werden können. Dementsprechend werden diese Modelle in [ZK99, KL00] zur Modellierung dieses Phänomens eingesetzt. Darüber hinaus finden insbesondere die verhältnismäßig einfachen Gibert-Elliot-Modelle mit zwei Zuständen auch als Modelle korrelierter Verluste in Rechnernetzen Anwendung (siehe z.B. [LZ06]).
Der Rest des Beitrags ist wie folgt gegliedert: Im Abschnitt 2 wird zunächst Lasttransformation definiert. Weiterhin rekapitulieren wir die für die folgenden Untersuchungen notwendigen Resultate zur Fragmentierungs-Transformation von BMAPs und schlagen Header-Transformationen für BMAPs vor. Im hierauf folgenden Abschnitt 3 wird eine Transformation auf *Batch Markovian Arrival Processes* zur Berücksichtigung von Auftragsverlusten vorgestellt. Darüber hinaus werden Formeln zur Berechnung des mittleren Durchsatzes von Ankunftsprozessen angegeben, die ihrerseits bereits (Paket-)Fragmentierung und Verluste modellieren. Die gemäß dieser Transformationen erzeugten Ankunftsprozesse werden im Abschnitt 4 mit simulativ transformierten Lasten verglichen, um die Genauigkeit der Transformationen zu überprüfen. Abschließend wird in Abschnitt 5 ein Fazit gezogen und ein kurzer Ausblick gegeben.

2 Lasttransformation

Innerhalb von Netzen von Bedienstationen ist die Last, wie sie an einer einzelnen Station anliegt, im Allgemeinen von der Diensterbringung anderer Stationen abhängig. Dies gilt im besonderen für gegenwärtige Rechnernetze, welche in Schichten und verbundenen Einzelsystemen organisiert sind. Jede Station beeinflusst hier die Last der jeweilig nachfolgenden Station, wie in Abbildung 1 schematisch dargestellt. Die so hervorgerufene Veränderung der Lastcharakteristiken wird im Folgenden als Lasttransformation von Primär- zu Sekundärlast bezeichnet. In der vorgenannten Abbildung können diese Transformationen beispielsweise Fragmentierung und Headergenerierung in einer *UDP*-Protokollinstanz oder Paketverlust bei der drahtlosen Übertragung modellieren.
Um diese Transformationen formal beschreiben zu können, wird Last, wie in Definition 1 [Wol99] dargestellt, definiert.

Definition 1 *Die* Last $L = L(E, S, IF, T)$ *wird definiert als eine Sequenz von Aufträgen, die während des Beobachtungsintervalls T an das Bediensystem S durch seine Umgebung E übergeben werden. Die Aufträge werden über die Schnittstelle IF übergeben, welche das Bediensystem von seiner Umgebung trennt.* ◊

Die Last kann somit durch eine Sequenz von Aufträgen a_i, die während des betrachteten Zeitintervalls T eintreffen, beschrieben werden. Für wohldefinierte Lasten sei der Ankunftsprozess definiert als Tupel aus

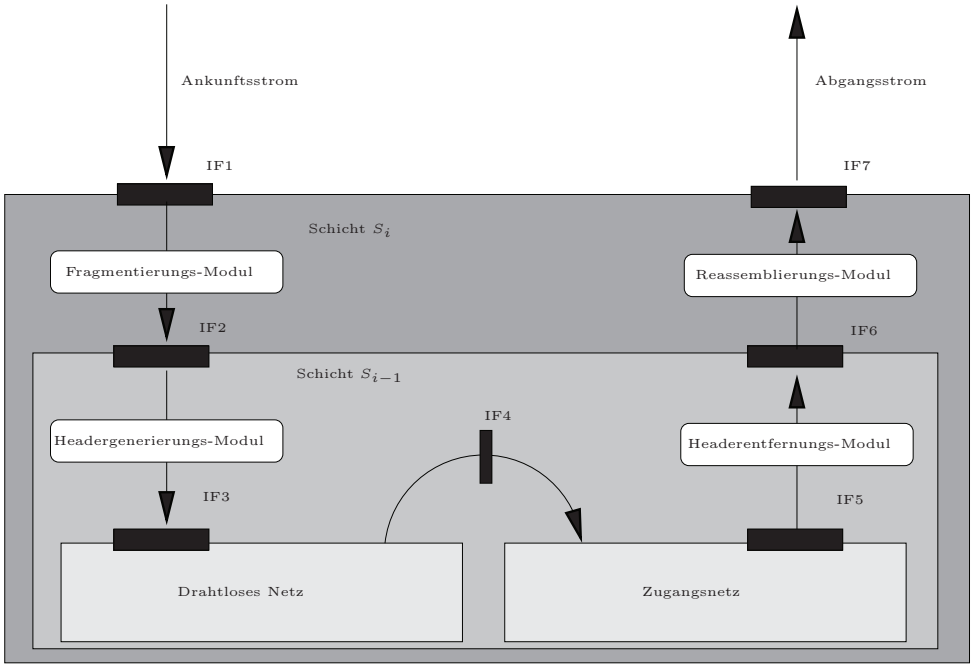


Abbildung 1: Übertragung eines Paketes als Ablauf von Transformationen

Ankunftzeitpunkten t_i und den Aufträgen a_i

$$\{(a_i, t_i) | a_i \in \mathcal{A}_i, t_1 \leq t_2 \leq ... \leq t_N, t_{1,...,N} \in T\} \tag{1}$$

Einzelne Aufträge a_i können hierbei beispielsweise Datenübertragungs- oder Verbindungsaufbauwünsche repräsentieren. Das vorgestellte Konzept ist weiterhin nicht auf Modellierung von Rechnernetzen beschränkt. Zur Modellierung von Datenbanksystemen könnten Transaktionen als Aufträge modelliert werden, um so zu einer Lastbeschreibung gemäß Definition 1 zu gelangen. Um eine detaillierte Spezifikation der Aufträge zu ermöglichen, verwenden wir darüber hinaus typisierte Aufträge sowie eine typabhängige Menge von Auftragsattributen. Aufbauend auf Definition 1 seien die folgenden 4 Klassen von Transformationen definiert:

1. Wir definieren eine Auftragstransformation als Abbildung T_A, welche eine Sequenz von Primärlastaufträgen $A^p = (a_1^p, ..., a_N^p)$ auf eine Sequenz von Sekundärlastaufträgen $A^s = (a_1^s, ..., a_K^s)$ für eine gegebene Lasttransformation abbildet.

$$T_A : A^p \rightarrow A^s$$

2. Die Transformation des zeitlichen Verhaltens sei als Abbildung der Ankunftszeitpunkte der Primärlast $T^p = (t_1^p, ..., t_N^p)$ auf die Ankunftszeitpunkte der Sekundärlast $T^s = (t_1^s, ..., t_K^s)$ definiert.

$$T_T : T^p \rightarrow T^s$$

3. Falls die Transformation von Attributen nicht unabhängig vom zeitlichen Verhalten erfolgen kann, so wird die Transformation der Aufträge beschrieben durch die Abbildung:

$$T'_A : A^p \times T^p \rightarrow A^s$$

4. Analog hierzu:

$$T'_T : A^p \times T^p \rightarrow T^s$$

Hiervon ausgehend ist es prinzipiell möglich, simulative Transformationen durchzuführen, was jedoch aus zwei Gründen nachteilig ist: Einerseits können bei einem simulativen Ansatz auch aus einem stochastischen Primärlastmodell nur deterministische Sequenzen konkreter Ankünfte als Sekundärlast gewonnen werden, indem Auftragssequenzen in Übereinstimmung mit dem Primärlastmodell erzeugt werden und

diese dann transformiert werden. Dies stellt eine Einschränkung der Allgemeingültigkeit des Modells dar. Andererseits ist Last, welche durch simulative Transformation erzeugt wurde, warteschlangentheoretischen Methoden unzugänglich, was insbesondere die Leistungsbewertung erschwert.
Um eine weitgehende analytische Handhabbarkeit der transformierten Modelle zu sichern, ohne gleichzeitig die Beschreibungsmächtigkeit der Modelle allzu sehr einzuschränken, werden im vorliegenden Beitrag Transformationen auf *Batch Markovian Arrival Processes* (BMAPs) untersucht [Luc93], wie sie in Definition 2 dargestellt sind.

Definition 2 *Ein* Batch Markovian Arrival Process *(BMAP) wird definiert als markovscher Ankunftsprozess mit infinitesimaler Generatormatrix:*

$$\mathbf{Q} = \begin{pmatrix} D_0 & D_1 & D_2 & D_3 & \dots \\ & D_0 & D_1 & D_2 & \dots \\ & & D_0 & D_1 & \dots \\ & & \vdots & & \ddots \end{pmatrix}$$

Weiterhin seien die $(m \times m)$ *Matrizen* $D_{0,\dots,\infty}$ *definiert durch:*

$$(D_0)_{ii} = -\lambda_i, \quad 1 \leq i \leq m$$

$$(D_0)_{ij} = \lambda_i p_i(0,j), \quad 1 \leq i,j \leq m \wedge i \neq j$$

$$(D_k)_{ij} = \lambda_i p_i(k,j), \quad 1 \leq i,j \leq m \wedge k > 0$$

$$\sum_{\substack{j=1 \\ j \neq i}}^{m} p_i(0,j) + \sum_{k=1}^{\infty}\sum_{j=1}^{m} p_i(k,j) = 1, \quad 1 \leq i \leq m$$

◊

Hierbei bezeichnet $p_i(k,j)$ die Wahrscheinlichkeit, dass beim Übergang von Zustand i in Zustand j eine Batch-Ankunft der Größe k erfolgt. Es können somit beliebige diskrete Verteilungen mit möglicherweise unendlichem Träger modelliert werden. Die Gesamtrate, mit der Zustand i verlassen wird, ist weiterhin gegeben durch λ_i. *BMAP*-Prozesse sind abgeschlossen gegenüber Aggregation, so dass auch Transformationen aggregierter Lasten oder Aggregationen transformierter Lasten angegeben werden können.
In [HW07] wurde bereits eine BMAP-Transformation für Fragmentierungsvorgänge vorgeschlagen. Da im Folgenden auch die Kombination mehrerer Transformationen betrachtet werden soll, ist die dort vorgeschlagene Tranformationsvorschrift in Definition 3 dargestellt. Hierbei wird angenommen, dass die Zwischenankunftszeit von Fragmenten τ_f konstant ist mit $\tau_f > 0$. Diese konstante Zwischenankunftszeit kann im Kontext kontinuierlicher Markov-Prozesse nur approximativ modelliert werden; durch die Einführung zusätzlicher Zustände kann jedoch – falls nötig – ein höherer Übereinstimmungsgrad erreicht werden.
Da durch die konstante Zwischenankunftszeit zwischen den Fragmenten τ_f auch die Zwischenankunftszeit zwischen dem letzten Fragment eines Paketes und dem ersten Fragment des nachfolgenden Paketes verändert wird, erfolgt eine Reduktion der mittleren Verweilzeit des Prozesses in den Zuständen $w_{i,j}$. Diese wird um die mittlere Zeit $E[T_i^F]$, die für die Fragmentierung benötigt wird, reduziert. Die Übergangsraten wurden so gewählt, dass die durch eine Paketankunft im Zustand i induzierte Anzahl Fragmente im Mittel mit der durch den transformierten Prozess generierten Anzahl übereinstimmen. Die durch den transformierten Prozess generierte Fragmentanzahl ist hierbei geometrisch verteilt. In [HW07] wurden weiterhin Methoden angegeben mit Hilfe derer auch andere Verteilungen modelliert werden können.
Jeder Zustand des untransformierten BMAPs resultiert in mindestens zwei Zuständen der transformierten BMAPs, wobei die Zustände $w_{i,j}$ die eigentlichen Transitionsraten des Primärlastmodells repräsentieren und die Fragmenterzeugung durch die Zustände g_i modelliert wird.
Hierbei steht $p_i^{TF}(k,j)$ für die Wahrscheinlichkeit, dass aus Zustand i ein Übergang in Zustand j erfolgt, der ein abschließendes Teilfragment der Länge k induziert. Weiterhin entspricht m_p der Anzahl der Zustände von B^P.[1] Die Anzahl der im Zustand i generierten Fragmente sei des Weiteren durch die Zufallsvariable L_i^F gegeben.

[1] Zur Verdeutlichung benutzen wir für die Anzahl der Zustände des Primär- bzw. Sekundärlastmodells die Bezeichnung m_p bzw. m_s.

Definition 3 *Sei die Primärlast definiert durch BMAP B^P. Die Sekundärlast nach der Fragmentierung mit maximaler Fragmentlänge M wird dann modelliert durch BMAP B_F^S mit der Menge der* erzeugenden Zustände [2]

$$G = \{g_i, 0 < i \leq m_p\},$$

der Menge der Wartezustände

$$W = \{w_{i,j}, \forall i, j.\exists k.p_i(j,k) > 0\}$$

und den Matrizen $D_{0,\ldots,\infty}$

$$(D_k)_{w_{i,j},g_j} = \begin{cases} \frac{\lambda_j}{1-\lambda_j \cdot E[T_i^F]}, & k = 0 \\ 0, & k \neq 0 \end{cases}$$

$$(D_k)_{w_{i,j},w_{i,j}} = \begin{cases} -\frac{\lambda_j}{1-\lambda_j \cdot E[T_i^F]}, & k = 0 \\ 0, & k \neq 0 \end{cases}$$

$$(D_k)_{g_i,g_i} = \begin{cases} \lambda_f \cdot \left(1 - \frac{1}{E[L_i^F]+1}\right), & k = M \\ 0, & k \neq M \wedge k \neq 0 \\ -\lambda_f, & k = 0 \end{cases}$$

$$(D_k)_{g_i,w_{i,j}} = \begin{cases} \lambda_f \cdot p_i^{TF}(k,j) \cdot \frac{1}{E[L_i^F]+1}, & k \leq M \\ 0, & k > M \end{cases},$$

wobei $\lambda_f = \tau_f^{-1}$.

◊

Neben der Fragmentierung stellt – insbesondere für Effizienzbetrachtungen – das Hinzufügen von zusätzlichen Informationen (*Header*) eine wichtige Transformation dar. Dies gilt insbesondere für verlustbehaftete Übertragungswege, weil dort die Fragmentierung großer Informationseinheiten einen Weg zur Reduktion der Fehlerhäufigkeit darstellt und sich diese nur unter Berücksichtigung des zusätzlichen Overheads realistisch betrachten lässt. Daher sei die Headertransformation für BMAPs definiert, wie in Definition 4 dargestellt.

Definition 4 *Sei die Primärlast definiert durch BMAP B^P mit den Matrizen $(D_i^P)_{j,k}$. Das Hinzufügen von h Aufträgen zu einer Batchankunft der Länge i ergibt BMAP B_H^S, welcher definiert ist durch die Matrizen*

$$D_0^S = D_0^P$$
$$D_{i+h}^S = D_i^P, \quad i > 0$$

◊

Der vorgenannten Transformation liegt dabei die Annahme zugrunde, dass Zustandsübergänge mit Ankünften der Größe 0 keinen Paketversand zur Folge haben – also auch kein Headerversand nötig wird. Weiterhin zieht jede Ankunft der Größe i den Versand eines Paketes der Länge $i + h$ nach sich, so dass die korrespondierenden Matrizen D_{i+h}^S und D_i^P identisch sind.

3 Verlusttransformation

Neben den beiden oben erwähnten Transformationen spielen insbesondere bei drahtlosen Übertragungskanälen Auftrags- bzw- Paketverluste eine große Rolle. Nachfolgend soll daher eine Transformation vorgeschlagen werden, welche die Berechnung des transformierten BMAPs sowohl für Fehlermodelle mit konstanter Fehlerrate als auch für markovsche Fehlermodelle erlaubt. Diese ist in Definition 5 dargestellt.

[2] Wir bezeichnen im Kontext der Fragmentierungstransformation diejenigen Zustände, bei deren Verlassen Batch-Ankünfte größer 0 mit Wahrscheinlichkeit größer 0 auftreten, als *erzeugende Zustände*.

Definition 5 *Die augenblickliche Verlustrate zum Zeitpunkt t werde durch den Markovprozess* $X = (X(t) : t \geq 0)$ *mit infinitesimaler Generatormatrix* Q_L *gesteuert; die Fehlerwahrscheinlichkeit im Zustand i für ein Paket der Länge j sei* $r_{i,j}$*. Dann ergibt sich aus BMAP* B^P *der verlustbehaftete BMAP* B_L^S *mit der Zustandsmenge* $Z = \{Z_B \times Z_X\}$ *und den Matrizen*

$$(D_0^L) = D_0^P \oplus Q_L + \sum_{i=1}^{\infty} E_i \otimes D_i^P$$

$$(D_i^L) = (I - E_i) \otimes D_i^P, \quad i \geq 1$$

Hierbei sei E_i *definiert als*

$$E_i = diag(r_{1,i}, ..., r_{m_L,i}) = \begin{pmatrix} r_{1,i} & 0 & \cdots & 0 \\ 0 & r_{2,i} & \cdots & 0 \\ \vdots & \cdots & \ddots & \vdots \\ 0 & \cdots & 0 & r_{m_L,i} \end{pmatrix}.$$

◊

Hierbei steht $\otimes$ für das Kroneckerprodukt zweier Matrizen, $\oplus$ für die Kroneckersumme (siehe z.B. [Lau05]), m_L für die Anzahl der Zustände von $X(t)$ und I für eine Einheitsmatrix entsprechender Größe. Im folgenden soll nun gezeigt werden, dass durch die Transformation die zeitlichen Eigenschaften der beiden Prozesse – also Ankunfts- und Fehlerprozess – erhalten bleiben. Hierzu seien auf den verlustbehafteten BMAPs zwei Äquivalenzrelationen $\sim_B$ bzw. $\sim_L$ definiert, die jeweils diejenigen Zustände umfassen, bei denen sich der Ankunfts- bzw. der Fehlerprozess im selben Zustand befinden. (Hierbei wird der endliche zweidimensionale Zustandsraum aus Def. 5 eindimensional betrachtet in der Ordnung wie sie durch die Kroneckeroperationen induziert wird.) Es wird sich zeigen, dass die Raten zwischen Zustandsmengen, die in verschiedenen Äquivalenzklassen enthalten sind, den Raten des korrespondierenden Ausgangsprozesses gleichen.

$$i \sim_B j \Longleftrightarrow i \equiv j \ (mod \ |Z_B|) \tag{2}$$

$$i \sim_L j \Longleftrightarrow \left\lfloor \frac{i}{|Z_B|} \right\rfloor = \left\lfloor \frac{j}{|Z_B|} \right\rfloor \tag{3}$$

Hierauf aufbauend steht $\nsim_L$ bzw. $\nsim_B$ für die Nicht-Äquivalenz zweier Zustände bezüglich der in den Formeln (2) und (3) eingeführten Relationen. Weiterhin sei die Rate zwischen zwei Zuständen i und j des Fehlerprozesses mit $\lambda_{i,j}^L$ bzw. die Rate zwischen zwei Zuständen i und j des transformierten Prozesses mit $\lambda_{i,j}$ bezeichnet.

Satz 1 *Für zwei Zustände i,j mit* $i \nsim_L j$ *bleiben die Übergangsraten des Fehlerprozesses zwischen den Äquivalenzklassen* $[i], [j]$ *durch die Verlusttransformation erhalten, d.h. es gilt:*

$$\lambda_{[i],[j]} = \lambda_{k,l}^L, \quad k = \left\lfloor \frac{i}{|Z_B|} \right\rfloor, \quad l = \left\lfloor \frac{j}{|Z_B|} \right\rfloor$$

Beweis Per Konstruktion gilt für zwei Zustände m,n, $m \sim_L i$, $n \sim_L j$, $m \sim_B n$:

$$\begin{aligned} \lambda_{m,n} &= \left(D_0^P \oplus Q_L\right)_{m,n} + \left(\sum_{i=1}^{\infty} E_i \otimes D_i^P\right)_{m,n} + \left(\sum_{i=1}^{\infty} (I - E_i) \otimes D_i^P\right)_{m,n} \\ &= \left(D_0^P \oplus Q_L\right)_{m,n} \\ &= \left(I \otimes D_0^P\right)_{m,n} + (Q_L \otimes I)_{m,n} \\ &= (Q_L \otimes I)_{m,n} \\ &= (Q_L)_{k,l} = \lambda_{k,l}^L \end{aligned}$$

Hieraus folgt:

$$\begin{aligned} \sum_{\forall m \sim_L i} \sum_{\forall n \sim_L j} \pi_m \cdot \lambda_{m,n} &= \lambda_{k,l}^L \sum_{\forall m \sim_L i} \pi_m \\ &= \lambda_{k,l}^L \end{aligned}$$

□

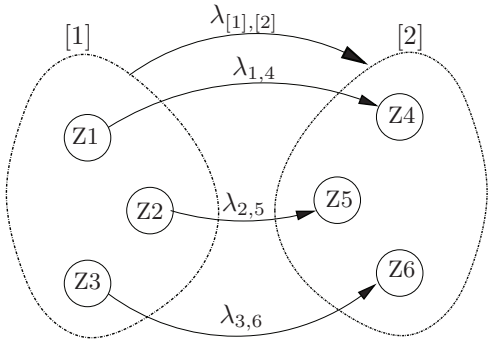


Abbildung 2: Raten zwischen Einzelzuständen und den korrespondierenden Äquivalenzklassen

Hierbei gibt π_m die Zustandswahrscheinlichkeit des Zustands m für den verlustbehafteten Ankunftsprozess an, unter der Bedingung, dass sich dieser in einem zu m äquivalenten Zustand (bezüglich $\sim_L$) befindet. Zur Verdeutlichung ist der Zusammenhang zwischen den Raten der Einzelzustände des transformierten Modells und den Raten zwischen den Äquivalenzklassen in Abbildung 2 exemplarisch dargestellt: Da die Transformation so vorgenommen wurde, dass die Raten zwischen den Einzelzuständen der Äquivalenzklassen [1] und [2] identisch sind und der Rate des Fehlermodells entsprechen, ist auch $\lambda_{[1],[2]}$ gleich dieser Rate. Dies entspricht dem Konzept der *Strongly Lumpable Markov Chains* (siehe z.B. [JB03]), wobei die Partionierung der Zustandsmenge jeweils entsprechend der Äquivalenzrelation durchzuführen ist.

Satz 2 *Für zwei Zustände i,j mit $i \nsim_B j$ bleiben die Übergangsraten des Ankunftsprozesses durch die Verlusttransformation erhalten, d.h. es gilt:*

$$\lambda_{[i],[j]} = \lambda^B_{k,l}, \quad k = i \bmod |Z_B|, \quad l = j \bmod |Z_B|$$

Beweis Folgt analog. □

Die Sätze 1 und 2 besagen insbesondere, dass durch die Kombination der beiden Markovprozesse die zeitlichen Eigenschaften sowohl von Ankunfts- als auch von Fehlerprozess erhalten bleiben. Da weiterhin die Übergangsraten zwischen den Äquivalenzklassen erhalten bleiben, entsprechen auch die jeweiligen stationären Wahrscheinlichkeiten denen der Ausgangsprozesse. Darüber hinaus wird im vorgeschlagenen Modell – aufgrund der Zeitlosigkeit der Batch-Ankünfte – davon ausgegangen, dass sich der Zustand des Fehlerprozesses während der Übertragung eines Fragmentes nicht ändert. Um den hierdurch induzierten Fehler nicht zu groß werden zu lassen, sollten die Änderungsraten nicht erheblich größer als die Übertragungsdauer eines Fragmentes sein (z.B. *Slow Fading*). Weiterhin muss bedacht werden, dass die vorgeschlagene Transformation in erster Linie auf Szenarien mit Vorwärtsfehlerkontrolle (*FEC*) anwendbar ist. Hierbei würden sich all diejenigen Bitfehler als Verlust im Sinne der Transformation niederschlagen, welche durch die Vorwärtsfehlerkontrolle nicht korrigiert werden können. Im Falle von Übertragungswiederholung (*ARQ*) kann diese jedoch ebenfalls verwendet werden, falls durch die eventuellen Wiederholungen nur mit geringer Wahrscheinlichkeit Rückstaus induziert werden. Ist dies der Fall, so müssten darüber hinaus die zeitlichen Verschiebungen, wie sie durch Übertragungswiederholungen hervorgerufen werden, in geeigneter Weise berücksichtigt werden.
Mit Hilfe der vorgeschlagenen Transformation lässt sich nun die Veränderung der Lasteigenschaften bis zur Ankunft im Zugangsnetz nachbilden (siehe Abbildung 1), wobei in Abhängigkeit vom vorliegenden Protokollstapel u.U. mehrfache Fragmentierungs- und Headertransformationen anzuwenden sind.
Der Effekt einer solchen Komposition von Transformationen ist in Abbildung 3 illustriert: Während in der linken Hälfte der Abbildung die Auswirkungen von Fragmentierungs-, Header- und Verlusttransformation auf Paketebene dargestellt ist, zeigt die rechte Hälfte der Abbildung die Kombination auf Prozessebene in schematischer Art und Weise. Hierbei sind die Zustände, welche die Prozesse zum jeweiligen Zeitpunkt

einnehmen, gemäß ihres Indexes (Z) dargestellt. $BMAP$ und Verlustprozess haben jeweils 2 Zustände und resultieren dementsprechend in einem transformierten Prozess mit 4 Zuständen. Bezüglich der Illustration auf Paketebene ist besonders auf die Tatsache hinzuweisen, dass nicht zwangsläufig sämtliche Fragmente eines Paketes verloren gehen müssen.

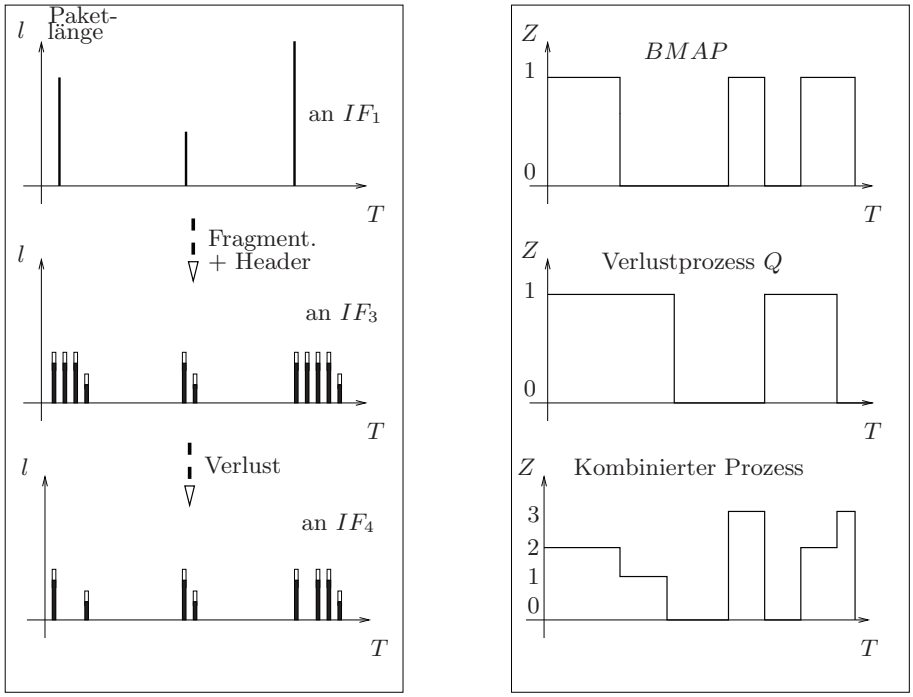


Abbildung 3: Illustration des Transformationsvorgangs auf Prozessebene und am Beispiel konkreter Auftragsankünfte (Bem.: Die genannten Schnittstellen IF_i (linke Graphik) beziehen sich auf Abb. 1.)

3.1 Berechnung der mittleren Rate

Da im vorliegenden Modell Paketankünfte als zeitlose Ereignisse angenommen werden, hängt die Wahrscheinlichkeit der Ankunft eines fehlerhaften Paketes ausschließlich von den stationären Zustandswahrscheinlichkeiten des Fehlerprozesses π^L ab. Hieraus folgt, dass sich die mittlere Rate des verlustbehafteten $BMAPs$, wie in Formel (4) dargestellt, angeben lässt, wobei e dem Spaltenvektor $[1\ 1\ ...\ 1]'$ entsprechender Länge entspricht. Weiterhin gibt π die stationären Zustandswahrscheinlichkeiten des BMAPs an, welcher die Primärlast beschreibt.

$$R = \sum_{i=1}^{\infty} i \left((\pi \cdot D_i \cdot e) \cdot (\pi^L \cdot E^i \cdot e) \right) \tag{4}$$

Erfolgt zusätzlich eine Fragmentierung der Ankünfte, so lässt sich die mittlere Rate des fragmentierten Verlustprozesses aufbauend auf der folgenden Überlegung angeben: Der fragmentierte Ankunftsprozess, welcher Paketverlusten ausgesetzt ist, besteht aus einer Abfolge von Ankünften der maximalen Batch-Größe M unterbrochen von Ankünften einer Batch-Größe kleiner M. Dies entspricht der Unterteilung eines Paketes auf Primärlastebene in $n-1$ Fragmente voller Länge, sowie einem Paket der Länge 0 bis $M-1$, welches die restlichen Daten des Paketes enthält.[3] Die Fragmente voller Länge erfahren eine Verlustwahrscheinlichkeit E^M, wobei pro Paket im Mittel $E[L^F]$ solcher Fragmente versendet werden. Entsprechend folgt:

$$R = \sum_{i=1}^{M-1} i \left((\pi \cdot D_i^{TF} \cdot e) \cdot (\pi^L \cdot E^i \cdot e) \right) + M \cdot \left((\pi \cdot \underline{\lambda}) \cdot (\pi^L \cdot E^M \cdot e) \right) \cdot E[L^F] \tag{5}$$

[3] Die Fragmentlänge 0 des $n-ten$ Paketes entspricht genau dem Fall, in dem die maximale Fragmentlänge die Paketlänge ohne Rest teilt und modelliert das "Nicht-Versenden" des Teilfragmentes.

Hierbei steht $\underline{\lambda}$ für den Zeilenvektor bestehend aus den Abgangsraten aus den Zuständen $1, ..., n$ und D_i^{TF} ist die Ratenmatrix der Teilfragmente der Länge i. Zur realistischen Betrachtung des Durchsatzes gilt es allerdings zu bedenken, dass einerseits jedes Fragment zusätzlichen Overhead induziert und andererseits eine Unterteilung in kleinere Fragmente grundsätzlich eine geringere Fehlerwahrscheinlichkeit zur Folge hat. Um diesen Randbedingungen gerecht zu werden, erfolgt die Erweiterung von Formel (5) um den Faktor $\frac{i}{i+h}$, so dass folgt:

$$R' = \sum_{i=1}^{M-1} i \left((\pi \cdot D_i^{TF} \cdot e) \cdot (\pi^L \cdot E^i \cdot e) \right) \cdot \frac{i}{i+h} + M \cdot \left((\pi \cdot \underline{\lambda}) \cdot (\pi^L \cdot E^M \cdot e) \right) \cdot E[L^F] \cdot \frac{M}{M+h} \quad (6)$$

Mit Hilfe von Formel (6) ist es jetzt insbesondere möglich, für gegebene Kombinationen von Ankunfts- und Fehlerprozess eine hinsichtlich des Gesamtdurchsatzes optimale Fragmentgröße zu ermitteln. Hierbei ist zu beachten, dass nicht mit jedem Zustandswechsel eine Neuberechnung dieses Wertes nötig wird, sondern nur die Prozesse als solche eine Rolle spielen. Weiterhin lässt sich erkennen, dass bezüglich des Ankunftprozesses in erster Linie die mittlere Größe der Primärlastankünfte von Bedeutung ist. Die eigentliche Verteilung wirkt sich nur im ersten Summanden von Formel (6) aus. Dessen Gewicht nimmt aber mit zunehmender Gesamtgröße ab. Weiterhin lässt sich für die Verteilungen der Teilfragmentlänge in vielen Fällen mit guter Näherung eine Gleichverteiltheit annehmen [HW07], so dass die eigentliche Paketlängenverteilung einen geringen Einfluss auf das Ergebnis der Summe hat.
Dies wird auch durch den in Abbildung 4 dargestellten Durchsatz in Abhängigkeit von der Fragmentgröße bestätigt. Wir betrachten das folgende Szenario: Als Fehlermodell fand dort ein *Gilbert-Elliot-Modell* Verwendung.[4] Ein solches Modell besteht aus zwei Zuständen, wobei im Zustand 1 eine geringe Fehlerrate r_1 angenommen wird, im Zustand 2 jedoch eine höhere Fehlerrate r_2. Als Fehlerraten pro Auftrag wurden $r_1 = 0$ und $r_2 = 10^{-3}$ gewählt. Die Generatormatrix Q_1 für das verwendete Fehlermodell G_1 ist in Formel (7) dargestellt.

$$Q_1 = \begin{pmatrix} -8 & 8 \\ 2 & -2 \end{pmatrix} \quad (7)$$

Weiterhin erzeugen alle betrachteten Primärlastmodelle 138000 Aufträge pro Sekunde, wobei der konkrete Auftragsprozess jeweils variiert wurde. BMAP B_1 erzeugt Pakete gleichverteilter Länge mit einer Rate von 1000 Aufträgen pro Sekunde, während BMAP B_2 ein einfaches Modell eines Videostromes darstellt, bei welchem mit einer Rate von 25 Aufträgen pro Sekunde in 3 Zuständen Pakete mit normalverteilter Länge erzeugt werden. BMAP B_3 erzeugt weiterhin Aufträge konstanter Länge mit einer Rate von 1000 Aufträgen pro Sekunde. Die mittlere Ankunftsrate der vorgenannten BMAPs in Abhängigkeit von der maximalen Fragmentlänge wurde nach Formel (8) berechnet. Hierbei erfolgte in Analogie zu Formel (6) die Erweiterung der Formel zur Ermittlung des mittleren Durchsatzes eines BMAPs (siehe z.B. [Luc93]) um den Faktor $\frac{i}{i+h}$ zur Gewichtung des Overheads.

$$\lambda^* = \pi \left(\sum_{i=1}^{\infty} \frac{i}{i+h} i D_i \right) e \quad (8)$$

In der linken Hälfte der Abbildung lässt sich erkennen, dass für alle 3 Ankunftsprozesse der optimale Durchsatz bei $M \approx 290$ erreicht wird und dass trotz der unterschiedlichen Charakteristiken alle Kurven einen ähnlichen Verlauf nehmen.
In der rechten Hälfte der Abbildung erfolgt der Vergleich zwischen dem Durchsatz der transformierten BMAPs λ^* und dem direkt berechneten Durchsatz R'. Es lässt sich (erfreulicherweise) kein Unterschied zwischen beiden Kurven erkennen.

4 Vergleich mit Simulationsresultaten

Neben dem analytischen Vergleich der mittleren Rate soll im Folgenden eine detailliertere Validierung der mittels Transformation erzeugten Ankunftsprozesse vorgenommen werden. Diese erfolgt mit Hilfe des Vergleichs von Auftragssequenzen, wie sie in Übereinstimmung mit transformierten BMAPs erzeugt wurden und simulativ transformierten Auftragssequenzen. Es soll somit gezeigt werden, dass unter Nutzung der oben dargestellten Transformationen realitätsnahe Simulationsszenarien mit hoher Genauigkeit nachgebildet werden können. Die Simulation erfolgte unter Zuhilfenahme des Simulationspaketes *ns*2, wobei das

[4] Die im Folgenden genutzten Parametrisierungen haben ausschließlich exemplarischen Charakter. Konkrete Parametrisierungen für reale Kanäle können beispielsweise den in Abschnitt 1 erwähnten Arbeiten entnommen werden.

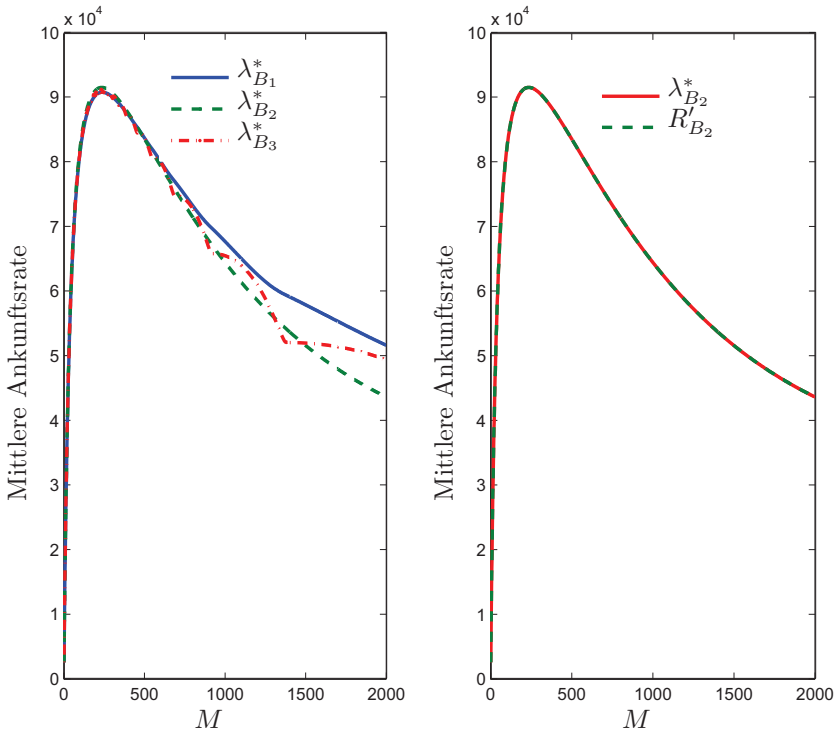


Abbildung 4: Mittlere Ankunftsraten λ^* der transformierten BMAPs $B_{1,2,3}$ und die direkt berechnete Rate R'_{B_2}

untersuchte Simulationsszenario in Abbildung 5 dargestellt ist: Die jeweilig betrachteten Ankunftsprozesse an der Schnittstelle zur Transportschicht (als vorausgesetzter Primärlastschnittstelle IF_P) wurden per UDP übertragen; die Fragmentierung der Pakete entsprechend der Maximalgröße M wurde innerhalb des *UDP-Moduls* vorgenommen. Die so fragmentierten Daten wurden simulativ über eine $100Mbit/s$-Leitung übertragen, auf der die Paketverluste induziert wurden. Die Messung der Ankunftszeiten und der Paketgrößen erfolgte, wie in Abbildung 5 dargestellt, an der Schnittstelle IF_S. Diese Schnittstelle repräsentiert die Schnittstelle oberhalb der Datensicherungsschicht (beobachtet in sämtlichen als Empfänger fungierenden Stationen/Rechnern).

In der Folge sollen dabei zwei Ankunftsprozesse mit abweichenden Charakteristiken, aber identischer

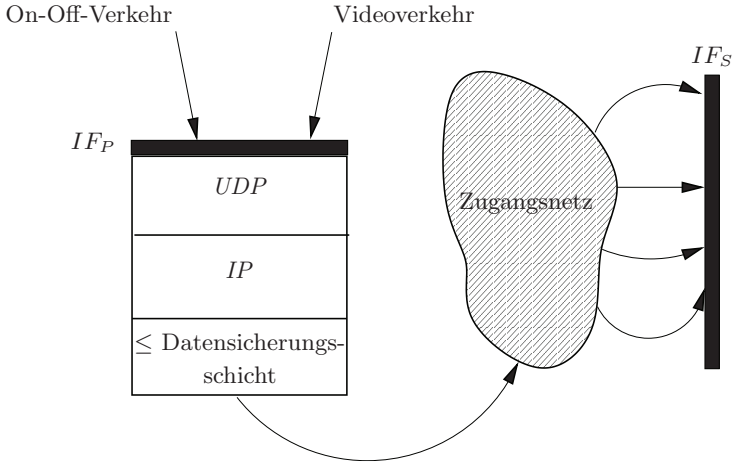


Abbildung 5: Übertragung eines Paketes in der Simulation

mittlerer Ankunftsrate, Gegenstand der Untersuchung sein. Diese sind:

- *Videoverkehr* B_2 Hierbei findet ein Modell Verwendung, in dem die 3 Rahmentypen eines MPEG-

Videos (siehe z.B. [Sym01]) durch jeweils eigene Zustände modelliert werden. Die Wahrscheinlichkeitsverteilung der Batch-Größen ist jeweils diskretisiert normalverteilt mit von einander abweichenden rahmenspezifischen Parametern.

- *On-Off-Verkehr* B_4 Dieses Modell verfügt über zwei Zustände mit einer mittleren Verweilzeit von jeweils $0,5s$. Im *On*-Zustand werden Pakete der Größe 2766 mit der Rate 100 erzeugt. Im Off-Zustand erfolgen keine Ankünfte.

Als Fehlermodell fand neben dem bereits in Abschnitt 3.1 verwendeten Modell G_1 das Modell G_2 mit abweichender Generatormatrix Q_2, siehe Formel (9), Verwendung.

$$Q_2 = \begin{pmatrix} -2 & 2 \\ 8 & -8 \end{pmatrix} \tag{9}$$

Die Fehlerprozess G_2 wurde hierbei wegen seiner vom Fehlerprozess G_1 stark abweichenden stationären Zustandswahrscheinlichkeiten gewählt. Weiterhin wurden neben den im Folgenden dargestellten Experimenten eine Vielzahl weiterer Experimente mit abweichenden Modellen und Startwerten des Zufallszahlengenerators durchgeführt, die indes bezüglich der Übereinstimmung der simulativ bzw. analytischen transformierten Prozesse vergleichbare Resultate lieferten. Anhand des On-Off-Modells B_4 wurde zunächst der Vergleich der erzeugten Gesamtlast vorgenommen. Da aufgrund der konstanten Paketlänge die Länge der Teilfragmente a priori bekannt ist, ist in Tabelle 1 jeweils die Zahl der Fragmente maximaler Länge und der Teilfragmente in Abhängigkeit der maximalen Fragmentgröße M und der Zustandswahrscheinlichkeit des fehlerbehafteten Zustands 2 angegeben. Hierbei entspricht $\pi_2 = 0,8$ dem Fehlerprozess G_1 bzw. $\pi_2 = 0,2$ dem Prozess G_2. Die Abweichungen sind durchgängig gering, wobei für die Modelle mit höherer Fehlerrate diesbezüglich eine leichte Zunahme zu beobachten ist. Geringere Abweichungen sind indes a priori zu erwarten, da im Verlustfall die Zwischenankunftszeit zwischen den angrenzenden Paketen nicht mehr exakt nachgebildet werden kann: Geht in der Simulation ein Fragment eines Paketes verloren, so beeinflusst dies nur die Zwischenankunftszeit der beiden angrenzenden Fragmente, die dann nicht mehr negativ exponentiell verteilt ist. Von dieser Tatsache muss jedoch in der Transformation aus Komplexitätsgründen abstrahiert werden, so dass besagte Zwischenankunftszeiten im Verlustfall weiterhin exponentiell verteilt sind.

Auch für den zweiten zu betrachteten Ankunftsprozess B_2 lassen sich sowohl Paketlängen als auch Zwischenankunftszeiten mit hoher Genauigkeit nachbilden.[5] Anhand der in Abbildung 6 dargestellten empirischen Verteilung der Fragmentlängen lässt sich erkennen, dass auch diese Verteilung von der Transformation mit hoher Genauigkeit nachgebildet wird – zwischen den beiden Kurven ist jeweils kein Unterschied erkennbar. Hierbei ist im linken Teil der Abbildung die Verteilung für den Fehlerprozess Q_1 dargestellt, während der rechte Teil die Verteilung für Q_2 zeigt. Weiterhin lässt sich erkennen, dass mit zunehmender Bitfehlerwahrscheinlichkeit die Wahrscheinlichkeit von Fragmenten mit maximaler Größe abnimmt, was auf die steigende Fehlerrate mit zunehmender Paketlänge zurückzuführen ist. Aus dem selben Grund kann mit zunehmender Bitfehlerwahrscheinlichkeit auch keine Gleichverteilung für die Länge der Teilfragmente mehr angenommen werden: Während im rechten Teil von Abbildung 6 der Verlauf für $X < 1500$ noch mit guter Näherung linear ist – was für Gleichverteiltheit spricht –, ist dies im linken Teil nicht mehr der Fall und kürzere Pakete sind deutlich wahrscheinlicher. Neben der Übereinstimmung der Längenverteilung

M	π_2	Simulation	Transformation
1000	0,8	177571/102529	178424/103478
1000	0,6	223380/226147	226147/124235
1000	0,2	313638/160041	312331/160004
1500	0,8	67854/76136	67747/76152
1500	0,6	94948/101194	95782/101367
1500	0,2	150640/152633	151454/153578
∞	0,8	45266	43841
∞	0,6	79264	78289
∞	0,2	146549	145374

Tabelle 1: Anzahl der ankommenden Batch-Aufträge maximaler/nicht maximaler Größe

[5] Den dargestellten Resultaten liegen eine Vielzahl von Simulationsläufen mit unterschiedlichen Zufallszahlengenerator-Startwerten und einer Simulationsdauer von $6000s$ zugrunde, welche alle nur sehr geringfügig voneinander abweichende Ergebnisse lieferten.

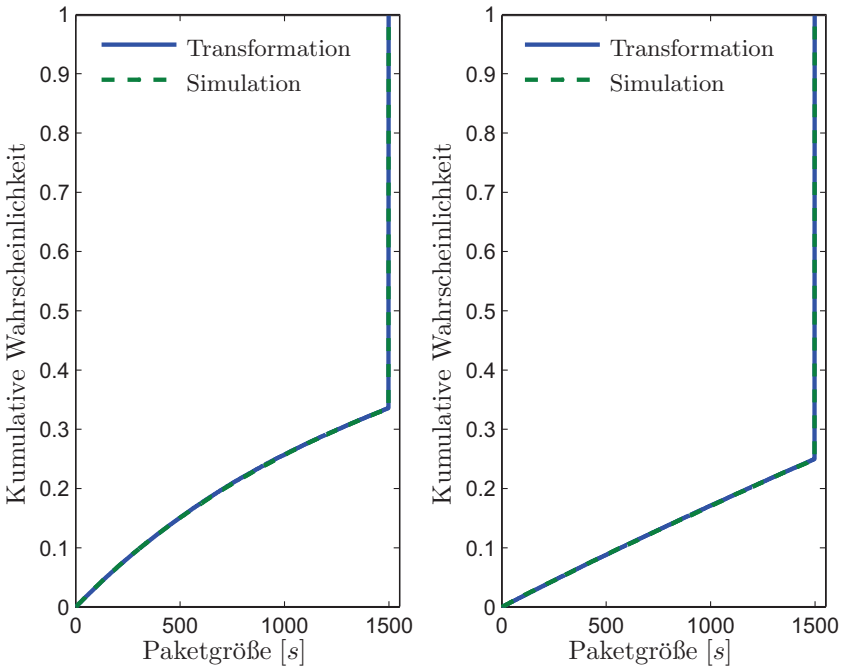


Abbildung 6: Kumulative Fragmentlängen-Wahrscheinlichkeit des verlustbehafteten Ankunftsprozesses bei maximaler Fragmentlänge $M = 1500$ für die Fehlerprozesse Q_1 (links) und Q_2 (rechts)

der erzeugten Fragmente ist die zeitliche Übereinstimmung von großer Wichtigkeit. Nach der analytischen Untersuchung dieses Aspektes in Abschnitt 3 soll hier nun am Beispiel von B_2 gezeigt werden, dass die vorgeschlagenen Transformationsverfahren darüber hinaus mit den simulativ gewonnenen Resultaten mit hoher Genauigkeit übereinstimmen, so dass auf eine sehr gute Realitätsnähe des analytischen Modellierungsansatzes geschlossen werden kann. Hierzu ist in Abbildung 7 die empirische Verteilung der Zwischenankunftszeiten für den Fehlerprozesse Q_1, wie sie jeweils simulativ und durch Transformation gewonnen wurden, dargestellt. Während in der linken Hälfte der Abbildung die Verteilung für $M = 1500$ zu sehen ist, zeigt die rechte Hälfte die Verteilung bei ausbleibender Fragmentierung. Für beide Szenarien bildet die Transformation die simulativ erzielten Ergebnisse mit hoher Genauigkeit nach. Geringe Abweichungen sind insbesondere für $M = 1500$ erkennbar. Dies steht im Einklang mit dem weiter oben erwähnten Problem der nicht-exponentiellen Verteilung der Zwischenankunftszeit im Verlustfall. Weiterhin ist zu beobachten, dass mit zunehmender Fehlerwahrscheinlichkeit die Wahrscheinlichkeit höherer Zwischenankunftszeiten erwartungsgemäß zunimmt.

5 Resümee und Ausblick

Dieser Beitrag beschäftigte sich mit dem Problem einer realitätsnahen Prädiktion von Lasten (im Sinne von Auftragssequenzen betrachtet an wohldefinierten Schnittstellen), ausgehend von unterschiedlichem (End-)Benutzerverhalten.
Dabei wurden die folgenden Ergebnisse erzielt: Es wurde aufgezeigt, wie durch das Konzept der Lasttransformation die Lasten an tieferen Schnittstellen einer Rechnernetzarchitektur (sog. Sekundärlastschnittstellen) ausgehend von bekannten Lasten an endbenutzernahen bzw. anwendungsorientierten Schnittstellen (sog. Primärlastschnittstellen) prognostiziert werden können. Eine Klassifikation von Lasttransformationen wurde vorgeschlagen, die u.a. berücksichtigt, ob der Auftragsankunftsprozess, die Auftragstypen und /oder die Auftragsattribute bei der entsprechenden Lasttransformation verändert werden. Im Zentrum des Beitrags stand die analytische Modellierung der Lasttransformationen “Headergenerierung”, “(Nachrichten-/Paket-) Fragmentierung” sowie “(Nachrichten-/Paket-) Verlust”, die allesamt auf Lastmodelle vom Typ BMAP (d.h. auf Markov’sche Ankunftsprozesse mit Gruppenankünften) angewendet wurden. Dabei wurden die Implikationen der entsprechenden Lasttransformationen untersucht sowohl im Hinblick auf den Auftragsabgangsprozess aus der betrachteten lasttransformierenden Komponente

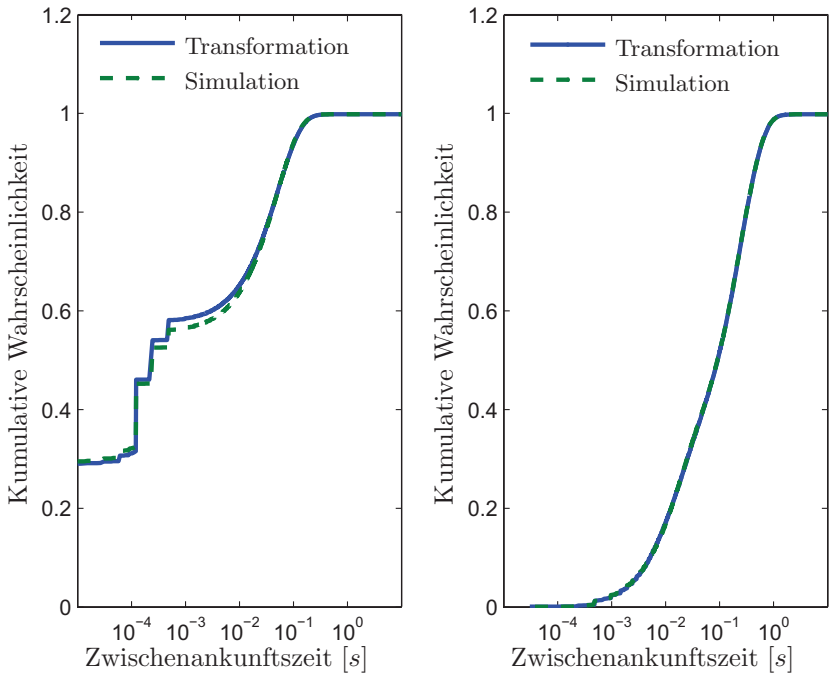


Abbildung 7: Kumulative Zwischenankunftszeit-Wahrscheinlichkeit des verlustbehafteten Ankunftprozesses bei Verwendung des Fehlerprozesses Q_1 für $M = 1500$ (links) und für ausbleibende Fragmentierung (rechts)

als auch in Bezug auf die abgehenden Aufträge selbst, z.B. Veränderung von deren Auftragsattributwerten (insbesondere die Auswirkungen auf die Längen – als zumeist zentrales Auftragsattribut – der zu übertragenden Daten bei Datenübertragungsaufträgen). Es gelang uns, die Auswirkungen der Lasttransformationen "Headergenerierung", "Fragmentierung" und "Auftragsverlust" auf die resultierenden Sekundärlasten weiterhin in Form von BMAP-Modellen zu beschreiben und diese, als Funktion der betrachteten Lasttransformation, zugleich geeignet zu parametrisieren. Unsere Simulationsstudien (auf *ns2*-Basis) zeigten auf, dass selbst hochkomplexe Lasttransformationen in realen Netzen – unter Annahme überlagerter Video- und Sprachübertragungen – noch auf realistische Weise durch die von uns vorgeschlagenen analytischen Lasttransformationen auf BMAP-Modellen berücksichtigt werden können. Hierbei bewährte sich u.a. auch unser Vorschlag, komplexe Lasttransformationen als Hintereinanderausführung elementarer Lasttransformationen zu modellieren. Überdies demonstrierte die in den Simulationsexperimenten betrachtete Fallstudie auch, wie Prognosen von Sekundärlasten auf Basis analytisch modellierter Lasttransformationen herangezogen werden können, um ein 'Tuning' freier Parameter des Netzes (wie z.B. eine im Netzbetrieb zu wählende maximale Fragmentgröße) in optimaler Weise vorzunehmen.
Unsere weiteren geplanten Arbeiten zur analytischen Modellierung von Lasttransformationen werden sich auf die Berücksichtigung weiterer Klassen von Transformationen beziehen, insbesondere auch auf Lasttransformationen, deren Verhalten noch stärker vom Netzzustand geprägt ist. Überdies streben wir eine vertiefte Modellvalidierung an, bei der direkte Messungen resultierender Sekundärlasten in exemplarisch gewählten Netzen (WLANs, Intranets, etc.) Verwendung finden sollen.

Literatur

[Bai99] G. Bai, Load measurements and modeling for distributed multimedia applications in high-speed networks, Ph.D. thesis, University of Hamburg, 1999.

[BC92] C. Blondia and O. Casals, Performance analysis of statistical multiplexing of VBR sources, IEEE INFOCOM '92 (Vol. 2) (Los Alamitos, CA, USA), IEEE Computer Society Press, 1992, 828–838.

[CW06] A. Chydzinski and R. Winiarczyk, Blocking Probability in a BMAP Queue, ISCC '06: Proc. 11th IEEE Symp. on Computers and Communications (Washington, DC, USA), IEEE Computer Society, 2006, 547–553.

[FC01] H.-W. Ferng and J.-F. Chang, Departure Processes of BMAP/G/1 Queues, Queueing Syst. Theory Appl. **39** (2001), no. 2-3, 109–135.

[Hof01] J. Hofmann, The BMAP/G/1 Queue with Level-Dependent Arrivals - An Overview, Telecommunication Systems **16** (2001), no. 3-4, 347–359.

[HSB04] T. Hofkens, K. Spaey, and C. Blondia, Transient Analysis of the D-BMAP/G/1 Queue with an Application to the Dimensioning of a Playout Buffer for VBR Video, NETWORKING, 2004, 1338–1343.

[HW07] S. Heckmüller and B.E. Wolfinger, Load Transformations for Markovian Arrival Processes, ASMTA 2007, June 2007, 35–43.

[JB03] R.W. Jernigan and R.H. Baran, Testing lumpability in Markov chains, Statistics and Probability Letters **64** (2003), 17–23.

[KL00] Y. Y. Kim and S. Li, Modeling multipath fading channel dynamics for packet data performance analysis, Wireless Networks **6** (2000), no. 6, 481–492.

[KLL03] A. Klemm, C. Lindemann, and M. Lohmann, Modeling IP traffic using the batch Markovian arrival process, Perform. Eval. **54** (2003), no. 2, 149–173.

[Lau05] A. J. Laub, Matrix Analysis for Scientists and Engineers, SIAM, 2005.

[LB06] H. W. Lee and J. W. Baek, Threshold Workload Control in the BMAP/G/1 Queue, International Conference on the Quantitative Evaluation of Systems (QEST), IEEE, 2006, 353–364.

[Luc91] D. M. Lucantoni, New results for the single server queue with a batch Markovian arrival process, Stoch. Mod. **7** (1991), 1–46.

[Luc93] D. M. Lucantoni, The BMAP/G/1 queue: a tutorial, Models and Techniques for Performance Evaluation of Computer and Communication Systems (L. Doniatiello and R. Nelson, eds.), Springer-Verlag, New York, 1993, 330–358.

[LZ06] H. Levy and H. Zlatokrilov, The effect of packet dispersion on voice applications in IP networks, IEEE/ACM Trans. Netw. **14** (2006), no. 2, 277–288.

[Neu79] M. F. Neuts, A versatile Markovian point process, J. Appl. Prob. **16** (1979), 764–779.

[Sym01] P. Symes, Digital Video Compression, McGraw-Hill, 2001.

[Wol99] B. E. Wolfinger, Characterization of Mixed Traffic Load in Service-Integrated Networks, Systems Science Journal **25** (1999), no. 2, 65–86.

[WZHB02] B. E. Wolfinger, M. Zaddach, K. D. Heidtmann, and G. Bai, Analytical modeling of primary and secondary load as induced by video applications using UDP/IP, Computer Communications **25** (2002), no. 11-12, 1094–1102.

[Zad01] M. Zaddach, Charakterisierung, Modellierung und Transformation von Videoverkehrslasten, Ph.D. thesis, University of Hamburg, 2001.

[ZK99] Q. Zhang and S. A. Kassam, Finite-State Markov Model for Rayleigh Fading Channels, IEEE/ACM Trans. Communic. **47** (1999), no. 11, 1688–1692.

[ZRM95] M. Zorzi, R. Rao, and L. Milstein, On the accuracy of a first-order Markov Model for data transmission on fading channels, Proc. IEEE ICUPC'95, November 1995, 211–215.

Macroscopic Workload Management and Mainframe Capacity Planning

Bruno Müller-Clostermann
Milen Tilev

University of Duisburg-Essen
Institute for Computer Science and Business Information Systems (ICB)
Schützenbahn 70, D-45117 Essen, Germany
{bmc|mtilev}@icb.uni-due.de

Abstract

Since in capacity planning activities the available information with respect to workload and system configuration is often rather coarse we propose a technique which is based on a “macroscopic” (high level) view. Based on the analysis of utilization data collected from mainframe monitoring we introduce a heuristic algorithm for the allocation of logical partitions (LPARs) to multiprocessors. Subsequently queuing theoretic formulas for M/M/m- and G/G/m-models can be used to check to which extent a proposed LPAR allocation will fulfil prespecified service levels. The approach is illustrated by data taken from real world scenarios.

Keywords: Capacity Planning, Mainframes, Performance, Queueing Models, Velocity

1 Introduction

We contribute to a capacity planning methodology for mainframe systems using a macroscopic view of a complete IT installation. A macroscopic description consists of a rather coarse workload model obtained from global measurements of system utilization, and a system description which is focussed on the number of processors and the processing capacity according to scaling tables for the system under consideration. These assumptions lead to a very abstract model on system level. Moreover, the results hold for long term “steady-state” behaviour and do not distinguish between service classes or even individual tasks. Such a simplified view on a complex and distributed system matches perfectly the high level view of a capacity planner.

We focus on nowadays mainframe systems where the workload is given by a set of LPARs (logical partitions) which are running on several systems. As a consequence capacity planning has to cope with the management of multiple workloads (here: LPARs) and their allocation to a server (here called CEC = Central Electronic Complex). For the purpose of capacity planning measurement data are typically available only on a rather macroscopic level, e.g. the 15 minute averages of consumed MIPS (Million Instructions per Second) over a certain time interval of some weeks or months. The feasible system configurations are also considered on a coarse level, i.e. we solely include the number and type of processors in combination with application specific benchmark results in form of a scaling table.

We propose an algorithm for the allocation of a set of LPARs to a number of CECs based on these workload and resource descriptions. Moreover additional “real-life” restrictions between LPARs are taken into account. Additionally we use queuing theoretic results for symmetric multiprocessor systems to evaluate the obtained LPAR allocations with respect to the so called execution velocity. An integration of these models into the capacity planning process will allow an efficient evaluation of many different model scenarios. For related work on capacity planning we refer to [Gunt00, Gunt07, MeAl02, MeAD04, MüCl01].

The rest of the paper introduces briefly into mainframe computing (Sec. 2) and into workload management and capacity planning (Sec. 3). Afterwards we introduce a heuristic algorithm for the

allocation of workloads to servers (Sec. 4) and quantitative evaluation of execution velocity by means of queueing theoretic formulas (Sec. 5).

2 Mainframe computing

2.1 Mainframe architecture

Mainframes belong to the most advanced high performance systems. They provide processing power for business-critical, high volume online transaction processing and database applications. As an example, the IBM z9 servers allow up to 54 processors (abbreviated as CPs = Central Processors or PUs = Processing Units) which are organized on 1 to 4 books in a single mainframe. The properties of such a system include the ability to add and activate processors, memory, and I/O connectivity while the server is running. A fault-tolerant communication ring interconnects books; processors on other books are available for activation to maximize the recovery capability in the case of processor failures. Similar redundancy techniques are employed with respect to memory, I/O, power and cooling. Various operating systems are supported such as Linux, z/OS, z/VM and z/VSE. For an overview cf. [FCSM04].

2.2 Workload and logical partitions

Mainframe workload is given by transactional workload and batch workload, which are both generated by partitions (LPARs). An LPAR is a virtualized computing environment which abstracts from all physical devices. LPARs are equivalent to physically separate servers with no connections. The typical number of logical partitions running on a single system is 10 to 60 depending on the type of mainframe and on the requested processing capacity of the partitions.

The processing capacity requested by a logical partition is described by the amount of MIPS (Million Instructions Per Second) which is necessary to run all applications with satisfactory service levels. The amount of MIPS to be provided is usually fixed by a contract between customer and provider. Since mainframe workload is relatively homogeneous, this simplified view of processing capacity quantified as MIPS is widely accepted as a useful performance measure. The installed (available) as well as the requested processing power are both described in MIPS.

2.3 Monitoring and measurements

Performance management metrics for mainframe systems include throughput, utilization, response times and execution velocity. The IBM Workload Manager (WLM) is a component of the z/OS operating system and the workload processed is classified by the WLM in distinct service classes. For each service class a set of performance goals – also called service levels – may be specified, which are used by the WLM to manage dynamically the complete workload. The WLM constantly monitors the system and adapts processing and allocation of resources like CPU-time to meet the performance goals [IBM05].

The Resource Measurement Facility (RMF) supports a set of functionalities and capabilities to measure and store collected performance data. RMF collects data concerning workload, resource utilization and other system activities. Typically processor utilization, I/O device activity and response times are determined during measurement periods of a specified length and are aggregated into performance reports. Also the fulfilment of specified performance goals is monitored for each service class. Workload management with the WLM includes the definition of performance goals and the setting of an „importance” (an IBM term) to each goal. The WLM decides how much resources, such as CPU and IO channels, should be provided to meet the performance goals. RMF data sampled over many days and for many partitions define time series for each logical partition.

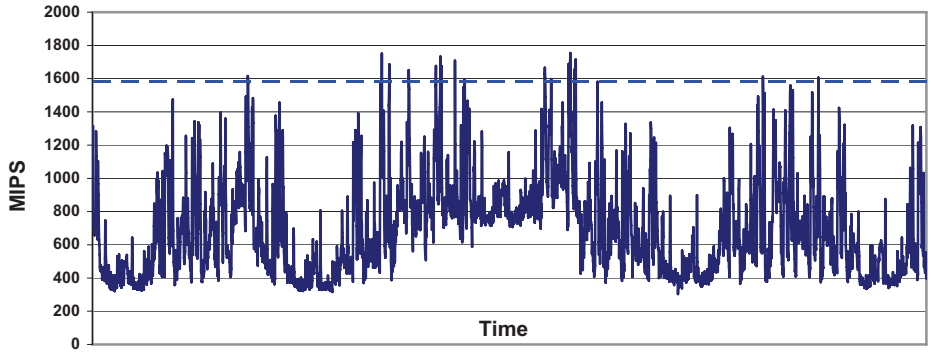


Fig. 1: Requested versus installed capacity over 4 weeks (Example a)

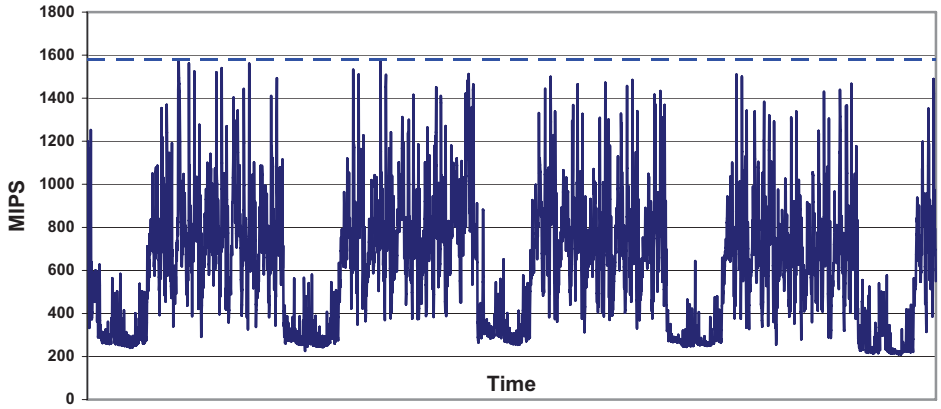


Fig. 2: Requested versus installed capacity over 4 weeks (Example b)

Fig. 1 and Fig. 2 show workload behaviours derived from empirical data as an example for the available workload description. The installed (=available) capacity for the logical partitions is displayed by the horizontal dotted line; the limit is defined by a contract and is set to 1600 MIPS in both cases. The curves show the requested (= consumed) capacity for a set of partitions over one month. Measurements of this type are usable for dimensioning and capacity planning and they are also useful to predict future QoS-violations which may occur due to short-term fluctuations of the workload.

3 Workload management and capacity planning

3.1 A capacity planning scenario

In this section we describe a typical real world scenario in mainframe capacity planning which comprises workload, system configuration and service levels. The level of detail and the input parameters follow the formulas known for M/M/m- or G/G/m-models to be used later. We focus on a scenario including a workload behaviour as a set of LPARs running on an m-way system (with m processors) providing a certain installed capacity (quantified in MIPS). We assume a future workload which is based on a forecast of LPAR behaviours and other assumptions with respect to shaping, capping, scheduling and priorization of online processing over batch.

The economic allocation of LPARs to CECs (servers) is a key to efficient and cost saving operation which has to consider many factors including accounting, contracting, costs for software licences and more. Here, the main objective is to achieve a trade-off between high utilization of the processor hardware and the prespecified service level requirements. An immediate consequence is to maximize utilization of the processors (per CEC) and at the same time to provide enough spare capacity in order to cope with peaks in the workload.

3.2 Workload model

The workload description consists of a mix of LPARs to be allocated on a system (= CEC). According to the given granularity of the workload description the processing time consumptions are assumed to be given in MIPS. We may distinguish several variants of workload characterization, e.g. the maximum of a 4 hour rolling average over all LPARs or a time series of 15 min values over all LPARs. In both cases the amount of requested service is quantified as an average value over a certain time interval.
An essential part of capacity planning is the development of workload forecasts. Mostly workload forecasts are based on historical data; seasonal effects and trends have to be considered as well as the evolution of the business processes. Since the logical partitions hosted on mainframes often serve a closed user community and are regulated by contracts between provider and user the variation of the workload is rather low and develops slowly over time. Moreover logical partitions can be moved between different mainframe systems to achieve appropriate workload profiles. Hence, workload forecasting is not a very hard task. For a further elaboration of this topic, we refer to [MeAD04, ch. 12], [BeMe04], [Gunt07].

3.3 Configuration model

A system configuration (for a single CEC) consists of the number of processors m and the corresponding amount of installed MIPS. Typically for each system or machine type the manufacturer or independent institutions provide tables with scaling factors. Often the MIPS-increments initially slowly decrease and finally approach a nearly linear behaviour. Of course system scalability of multiprocessor systems depends on the machine architecture, the operating system, the kinds of applications, the scheduling and workload management, and more. For a thorough discussion of scaling of multiprocessor systems see [Gunt00, Gunt07]. The scaling function is obtained from real benchmark data and shows a piecewise approximate linear behaviour with rather constant increments. Hence the usage of G/G/m formulas is justified for a restricted range of $n \pm 3$ processors.

3.4 Service levels

The definition of service levels requires numerical values for performance measures like throughput and response time. Here we employ the measure average velocity and give an outlook to average wait time and wait time percentiles. As an example we could define a service level like average velocity *E[V] = 80%*: The interpretation is that 80% of all tasks are served without any wait delay, and only 20% of all tasks have to suffer a certain wait time.

4 Workload allocation

4.1 The LPAR allocation problem

An approach (as described in [CapS07]) comprises as a first step an analysis of all LPARs with respect to the processing time consumption over a certain period of time followed by a grouping which is favourable with respect to utilization and available spare capacity. This LPAR allocation algorithm uses statistical data of each LPAR (mean, maximum, variance of consumed processing capacity), ordered capacity (i.e. amount of MIPS), type and number of processors, main memory, and more.

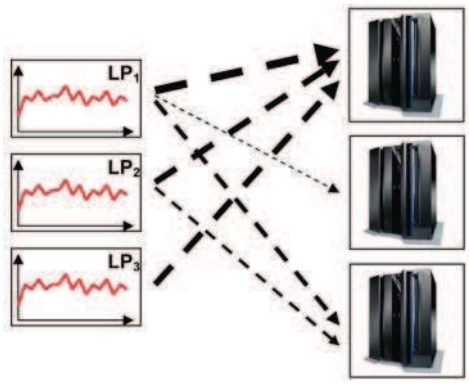


Fig. 3: Allocation of LPARs to CECs (basic outline)

As a result we obtain a value which quantifies the affinity of a LPAR towards a certain CEC. The capacity planner may choose from different strategies to control the allocation of LPARs; an example is the “maximum headroom strategy” which minimizes the maximum overall utilization on the CEC under consideration. The allocation process of LPARs to CECs and a possible result is briefly sketched in Fig. 3. The arrows indicate the strength of affinity to a certain CEC.

Fig. 4 displays for an LPAR allocation how the utilization curve may evolve over a certain time period. The horizontal line labelled C1 indicates the capacity limit which should normally suffice for a performant operation, line C2 indicates the installed (and active!) capacity, i. e. the difference C2-C1 is the available headroom (fast spare capacity) which will suffice to deal with temporary workload peaks. In case the COD option (COD = Capacity On Demand) is available we may care for extra capacity (slow or quiet spare capacity) which defines another line C3. The specification of concrete values for C1, C2 and C3 depends on many factors. In Fig. 4 the capacity limits C1 and C2 have been chosen in a way that a given hypothetical workload will be processed with just a few violations of the limit C1 and no violation of limit C2 at all. Of course we are well aware that a workload may evolve completely differently over time if more or less processing capacity is allocated; moreover the workload manager will take action in order to meet all performance goals. Nevertheless the capacity planner has to cope with his/her restricted macroscopic view on the overall system. Of course a model based approach which allows to quantify performance measures and to derive concrete values for C1 and C2 is highly desirable.

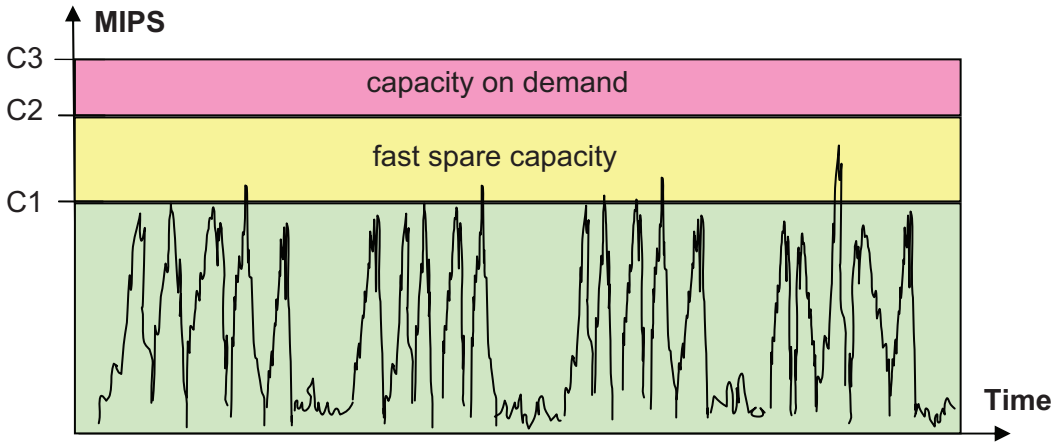


Fig. 4: Utilization over 4 weeks (basic outline)

As an example how real system behaviour is considered as part of the LPAR allocation strategy we refer to Fig. 5, where workload utilization behaviour is displayed before and after the application of shaping [CapS07]. The violations of the capacity limit (left diagram) have disappeared after shaping (right diagram).

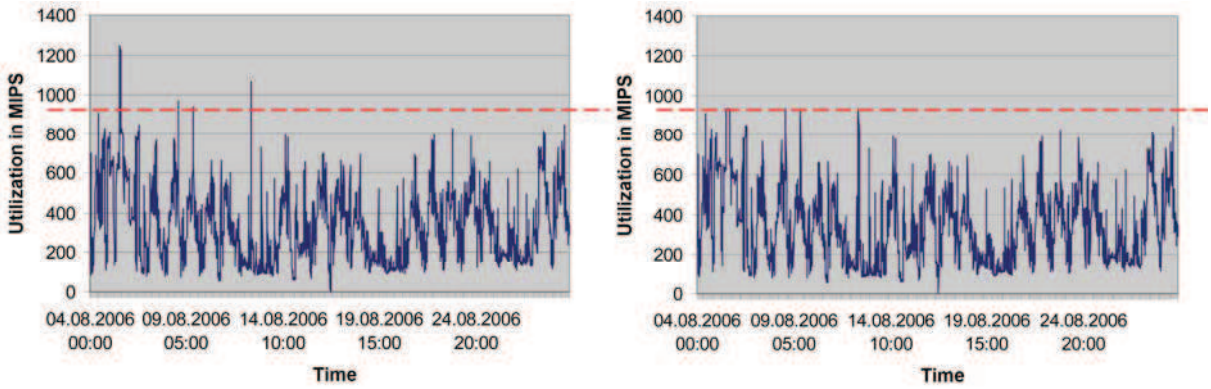


Fig. 5: Before and after workload shaping

Note that shaping is an approximate and heuristic technique that models the capping feature as applied during real mainframe operation, e.g. in order to restrict batch load to a specified maximum MIPS consumption.

4.2 A heuristic LPAR allocation algorithm

Our approach for finding suitable combinations of LPARs and CECs as sketched in Fig. 3 is based on a greedy heuristic algorithm, similar to the one introduced in [USR02]. The input for this algorithm is given as a set of LPARs and their associated properties like ordered MIPS, memory requirements, a time series of utilization measured in consumed MIPS, a set of CECs with their capacities given by the installed CPs and their types, amount of physical memory, and current LPARs residing on them. Additionally, restrictions are taken into account such that either some LPARs may not be placed together on the same CEC (in such a case, for two LPARs A and B we state that “A excludes B” or vice versa) or they have to be allocated on the same CEC.

To find the most suitable CEC for a certain LPAR, we define an abstract affinity function, which assigns each (LPAR, CEC)-pair a certain value quantifying an affinity value. The higher the affinity value the better the matching between LPAR and CEC. This function returns a special reserved value (NO AFFINITY) for pairs that are not acceptable at all. This is e.g. the case for pairs causing violations to one of the aforementioned capacity levels C1 or C2, or would result in forbidden allocations of two LPARs on the same CEC. Concrete implementations will compute the affinity value as a function of different criteria. Note that the affinity computation for a certain pair always takes into account all LPARs already associated with the CEC. This way it is possible to determine whether there is enough free capacity for allocating the LPAR under consideration.

Given these presumptions, the heuristic LPAR algorithm works as follows:

1. Group LPARs which belong together to LPAR clusters.
2. Compute the affinity value for each pair of a LPAR (or cluster) and a CEC; discard any pairs having the value NO AFFINITY.
3. Sort all LPARs and clusters in ascending order according to the number of pairs they participate in; for LPARs or clusters with equal number of pairings, use a second user definable criterion for obtaining the final sort order (e.g. mean utilization over the observed time interval, etc.).
4. Iterate over the sorted LPARs and clusters assigning the current one to the CEC it has the highest affinity value to; if no suitable CEC is available, then continue with 5; otherwise update the affinity values of any other LPARs and clusters, which participate in a pair with the just chosen CEC, as its state most certainly has changed. Resort the list of LPARs and clusters again by returning to step 3.
5. In case the algorithm terminates in step 4 without finding an appropriate allocation for every LPAR, we increase the capacity (number of CPs) on a certain user configurable CEC and restart the algorithm.

For the purpose of real world capacity planning scenarios, we decomposed the affinity function in two subfunctions: one checking any mandatory preconditions common to any scenario, and another user definable one, expressing different goals the capacity planner targets at. We call the later one an *assignment strategy*. In case a precondition is not satisfied for a certain (LPAR, CEC)-pair, we can discard it immediately without computing the user defined assignment strategy value. Our current mandatory preconditions are as follows:

1. Maximum number of LPARs per CEC: In case the addition of the new LPAR or LPAR cluster results in a greater overall number of LPARs on the CEC, we discard the considered pair
2. Amount of available physical memory on the CEC vs. memory requirements of the new coming LPAR or cluster
3. Exclusions between LPARs already associated to the CEC and the new LPAR or LPAR cluster
4. Capacity levels C1 and C2 are not violated.

We have implemented several assignment strategies each gaining different results when used on the same sets of LPARs and CECs [CapS07]. Currently the most appropriate strategy for a certain capacity planning scenario has to be determined interactively.

An LPAR allocation obtained as the result of this algorithm delivers an average MIPS value which must be provided by an associated CEC. This value is called ordered or requested MIPS value. Additionally we have the installed MIPS value of the CEC; the utilization is defined as the ratio of requested MIPS and installed MIPS. Moreover we have the number of processors. These are the parameters necessary for a queuing model of type *M/M/m* or *G/G/m*.

5 Macroscopic performance modelling

5.1 Building a G/G/m model

Here we map the scenario sketched above to G/G/m-formulas which will provide quantitative results for average velocity, average wait time and wait time percentiles. As necessary parameters we have the number of processors m and the utilization ρ, both from the allocation algorithm in the preceding section. To keep the model as simple as possible we normalize it by setting the mean service time to *E[S] =1*. Under this assumptions we may easily determine the average velocity by using the Erlang-C formula (cf. sec. 5.2). For the calculation of average wait times additionally the coefficients of variation C_A and C_S are necessary which account for the variability of inter arrival times and service times. Since the approximate G/G/m-formula due to Allen/Cunneen includes the factor $z = C_A^2 + C_S^2$, we may refer to *z* as an aggregate variable which accounts for the overall workload variability.

Processing Units m	Installed MIPS	Requested MIPS	Utilization [%]
16	6211	6208	99,9
17	6596	6208	94,1
18	6984	6208	88,9
19	7372	6208	84,2
20	7760	6208	80,0
21	8148	6208	76,2
22	8563	6208	72,5

Table 1: Installed MIPS, requested MIPS and utilization

For a concrete capacity planning scenario we assume a single CEC which can be equipped with a certain number of processors. The range of interest for the number of processors here is m = 16, 17 …, 22, yielding the values for installed computing power as displayed in Table 1, cf. the column “Installed MIPS”. Furthermore we have allocated a set of LPARs to this CEC. The requested computing capacity is here 6208 MIPS cf. the column “Requested MIPS”. The column “Utilization” is given by the ratio *100*(Requested MIPS/Available MIPS) [%]* yielding utilizations from 72,5% to the extreme (non-.realistic) 99,9%.

So far we have concrete values for number of processors m and the utilization ρ. The workload variability *z* will be addressed later in sec. 5.4.

5.2 Average execution velocity

In the context of mainframe terminology the term execution velocity denotes a performance measure; it describes the ratio for the total amount of workload units that is accepted to be delayed due to waiting for system resources. Execution velocity is a number between 0 and 100, where the value 100 means that during the observation interval the workload did not encounter any wait delays, and the value 0 expresses that all work is delayed over the measurement interval, cf. the documentation of the workload manager [IBM 05]. The average velocity *E[V]* is a stationary (global) measure which is

calculated from the basic parameters ρ and m, whereas during system operation the execution velocity is determined from measurements over rather short observation periods.

Here we show how to calculate the average execution velocity *E[V]*, as an overall long-term measure of congestion in multi-server systems. No service classes are distinguished, i.e. all tasks are aggregated into a single workload class.

We use a well known formula from queuing theory to derive expressions for average execution velocity *E[V]*, *E[V]* is defined as *E[V]=100·(1-P[W>0])*%, where the waiting probability P_m=*P[W>0]* can be calculated using the Erlang-C-formula. *P[W>0]* is also named delay probability or wait probability and is denoted equivalently (in percent) as *100*P[W>0]%*. The classic formula for computing this wait probability is the Erlang-C formula or M/M/m/m-formula, [BGMT98, cf. (6.28)] or [Klein78]. The Erlang C-formula provides the probability that the number of customers K reaches or exceeds the limit m.

$$P[W > 0] = P_m = P(K \geq m) = \frac{(m\rho)^m}{m!(1-\rho)} \cdot \pi_0,$$ where the probability π_0 for the empty system is defined as

$$\pi_0 = \left[\sum_{k=0}^{m-1} \frac{(m\rho)^k}{k!} + \frac{(m\rho)^m}{m!} \frac{1}{1-\rho} \right]^{-1}.$$

As an illustrative example we display in Fig. 7 the average velocity *E[V]=100·(1-P[W>0])*% for *m* processors, *m = 1, 4, 8, 16, 32, 64*.

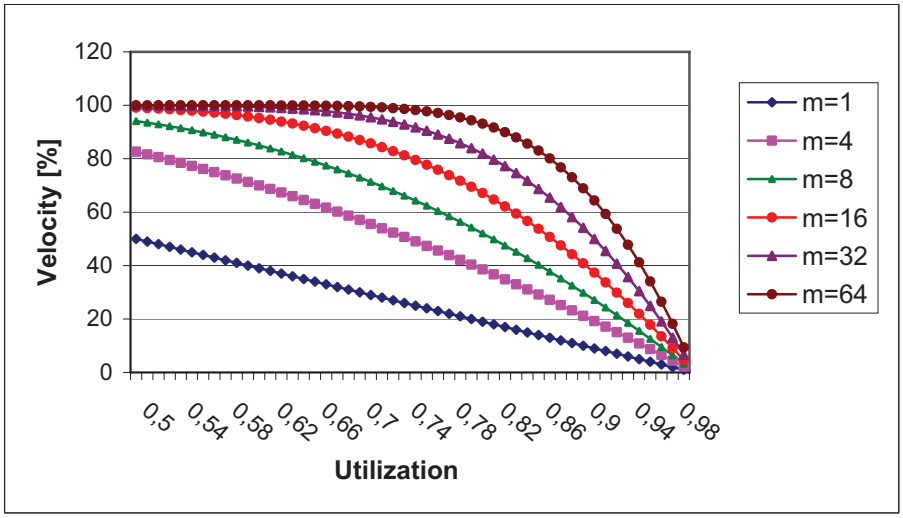


Fig. 6: Average execution velocity for m processors (m=1, …, 64)

Consider e.g. a value of 40% as an example for a possible service level. Fig. 6 shows under which utilization the average velocity drops below 40%, e.g. for *m=32* this is the case for utilizations beyond 92%. Note that the average execution velocity is not sensitive with respect to the variability of the arrival or service process, i.e. the coefficients of variation C_A and C_s do not influence the average velocity.

5.3 Examples for average velocity

If we apply the velocity formula to the values for utilization and number of processors in Table 1, we obtain the diagram displayed in Fig. 7. If we define a service level for average execution velocity of *E[V]=70%*, the diagram shows that a 20-way configuration should be appropriate. From Table 1 we see that a value of *E[V]=70%* corresponds to a utilization of 80% providing a headroom of 20%.

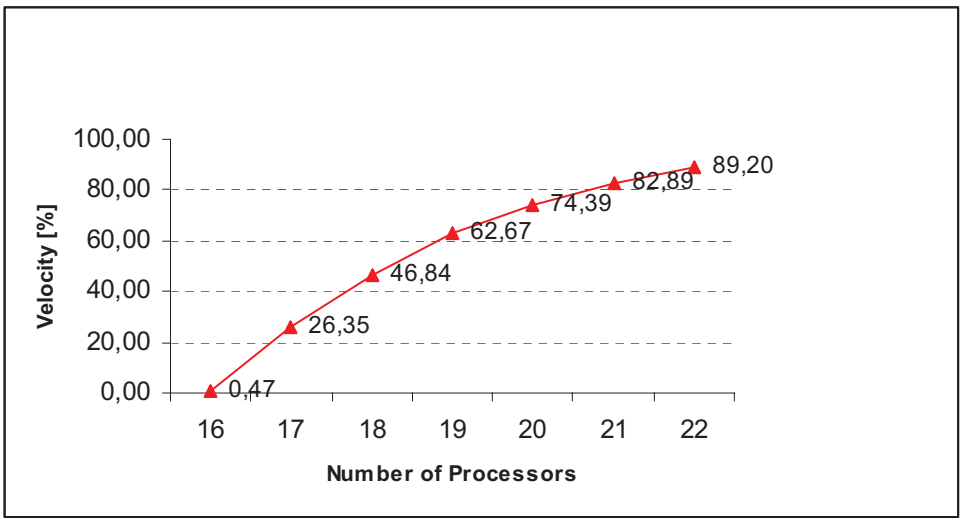


Fig. 7: Average velocity [%] versus number of processors (model result)

Fig. 8 shows a different type of diagram which has been obtained from real data (average utilization over 15 minute intervals) collected during one month from mainframe operation (IBM system z with 15 processors of type 2094).The execution velocities which have been calculated from the measured utilization time series are displayed as a frequency diagram.

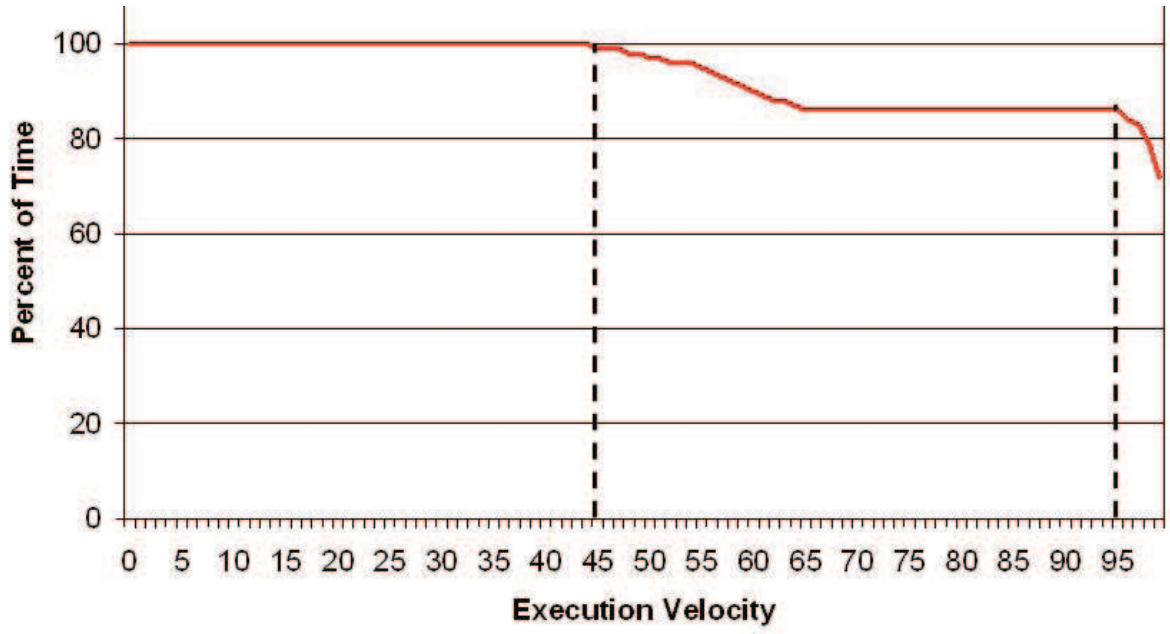


Fig. 8: Fulfilment of execution velocity over one month (from empirical data)

During 100% of the time intervals under consideration the execution velocity was at least 45%, and during about 85% of time it was better than 95% (see dashed vertical lines).

5.4 Average wait time and wait time percentiles

Apart from the average value we may additionally include the workload variability which is defined as $z = C_A^2 + C_S^2$. In case of well behaving workloads we may use the assumption z = 1.0 which may be reasonable and not too optimistic in many cases. Even values less than 1.0 may be justified in case of a good balanced system where automatic workload management by scheduling, capping and shaping cares for homogeneous workload behaviour.

The (normalized) average wait time may be evaluated using the formula of Allen/Cunneen [BGMT98]. Here it is displayed in combination with the probability P_m given above and the normalization *E[S]=1*:

$$E[W_{G/G/m}] = \overline{W} \approx \frac{P_m}{1-\rho} \cdot \frac{C_A^2 + C_S^2}{2m} = \frac{P_m}{1-\rho} \cdot \frac{z}{2m}.$$

Unlike the average velocity the average wait time is very sensitive with respect to the workload variability *z*. In particular for a highly variable workload (*z*>>1) and a utilization ρ exceeding 90% the wait time increases to extremely large values. As a result we can obtain characteristic curves as function of processor speed, number of processors, utilization and workload variability; examples can be found in [MüCl07].

Moreover, if we postulate that the wait time is exponentially distributed, the (inverse) mean wait time can be used as a parameter for the exponential wait time distribution. As a consequence the calculation of percentiles for wait time and response time is straightforward [MüCL07]. The wait time percentiles shown in Fig. 9 have been computed under usage of Table 1 and the assumption of moderate workload variability.

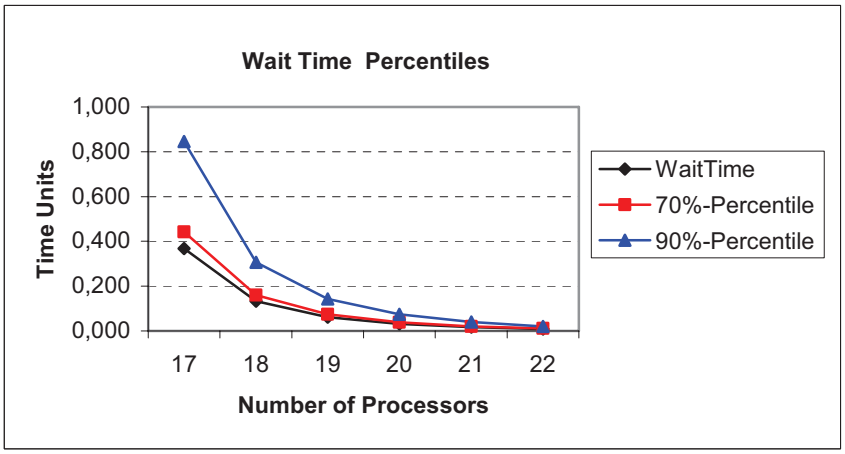


Fig. 9: Wait time percentiles for moderate workload variability (z=1.0)

The configurations with 18 and 19 processors both perform well with this type of workload. If we increase *z* the 18-way configuration will degrade, e.g. for *z*=5 the 90%-wait-time-percentile will exceed the value 1.5, i.e. wait time exceeds service time considerably.

The practical usage of the measures average wait time and wait time percentiles requires a meaningful estimation of the workload variability and subsequent validations.

6 Conclusion and future work

A very high level “macroscopic” view of multi-server workload management and capacity planning has lead to a heuristic algorithm for workload allocation. Formulas from queuing theory deliver useful measures like average execution velocity, average wait time and wait time percentiles to check the resulting scenarios with respect to prespecified service levels. The approach has been illustrated by data taken from real world scenarios.

For the purpose of capacity planning it is an obvious advantage to employ the metric “velocity” instead of the simplistic “utilization”. If additionally meaningful quantifications of the overall workload variability can be obtained also the average wait time or even wait time percentiles may be included in the capacity planning process.

Future work will integrate this approach into a capacity planning framework of a large service provider [CapS07]. Practical applications and field studies must be performed to demonstrate the feasibility of this approach. Hence the research focus will be on work load analysis in order to improve workload characterisation and model validation.

7 References

[BGMT98] Bolch, G., Greiner, S., de Meer, H., Trivedi, K.: Queueing Networks and Markov Chains – Modeling and Performance Evaluation with Computer Science Applications. John Wiley and Sons, 1998

[CapS07] CapSys User Manual. University of Duisburg-Essen, ICB, Systems Modeling Group, Internal Document, 2007

[FCSM04] Fair, M.L., Corkin, C.R., Swaney, S.B., Meany, P.J., Clarke, W.J., Alves, L.C., Modi, I.N., Freier, F., Fischer, W., Weber, N.E.: Reliability, availability, and serviceability (RAS) of the IBM eServer z990. IBM Journal Res. & Dev. (2004), vol 48, no. 3/4, 519-534

[Gunt00] Gunther, N. J.: The Practical Performance Analyst. Author Choice Press, 2000

[Gunt07] Gunther, N. J.: Guerrilla Capacity Planning - A Tactical Approach for Highly Scalable Applications and Services, Springer, 2007

[IBM05] IBM red books: System Programmer's Guide to: Workload Manager - Workload Manager Overview and Functionalities, Sept 2005.

[IBM06a] IBM red books: http://www.redbooks.ibm.com/, IBM Red books, 2006

[IBM06b] IBM z series: www.ibm.com/systems/z/, IBM Mainframes and the new system z9, 2006

[Klei78] Kleinrock, L.: 1978. Queueing Systems, Vol. II: Computer Applications, John Wiley, 1978

[MeAl02] Menascé, D. A., Almeida, V. A.: Capacity Planning for Web services – Metrics, Models, and Methods. Prentice Hall, 2002

[MeAD04] Menascé, D. A., Almeida, V. A., Dowdy, L.: Performance by Design: Computer Capacity Planning by Example. Prentice Hall, 2004

[MüCl01] Müller-Clostermann, B.: Kursbuch Kapazitätsmanagement - Kompendium für Planung, Analyse und Tuning von IT-Systemen. ISBN 3-8311-2823-5 (in German as pdf-version available under http://sysmod.icb.uni-due.de/), Essen, 2001

[MüCl07] Müller-Clostermann, B.: Using G/G/m-Models for Multi-Server and Mainframe Capacity Planning, ICB Research Report 16, www.icb.uni-due.de/researchreports, Essen, 2007,

[USR02] Urgaonkar, B., Shenoy, P., Roscoe, T.: Resource Overbooking and Application Profiling in Shared Hosting Platforms, Proceedings of the Fifth Symposium on Operating Systems Design and Implementation (OSDI '02), December 9 – 11, 2002, Boston, MA, USA

Ein Lastmodell für die Definition von I/O-Lasten beim Zugriff auf Sekundärspeicher paralleler Systeme

Daniel Versick, Heiko Kopp und Djamshid Tavangarian

Lehrstuhl für Rechnerarchitektur, Institut für Informatik,
Universität Rostock
Albert-Einstein-Str. 21, D-18059 Rostock
[daniel.versick | heiko.kopp | djamshid.tavangarian]@informatik.uni-rostock.de

Die Leistungslücke zwischen Prozessoren und Massenspeichern ist einem stetigen Wachstum unterzogen, weshalb eine fundierte Analyse und begründete Vorhersage der I/O-Leistung von Massenspeichern durch geeignete Benchmarks, die den Durchsatz und die Latenz von Sekundärspeicherzugriffen vermessen, immer größere Relevanz erhält. Insbesondere können Performance-Beschränkungen beim Zugriff auf den Sekundärspeicher identifiziert werden. Speziell in Hochleistungscomputern, die auf immer größere Datenmengen zugreifen müssen, ist eine Optimierung der Ressourcennutzung von besonderer Bedeutung. Da aktuell verfügbare I/O-Benchmarks typischerweise statisch definierte synthetische Zugriffsmuster benutzen, sind ihre Ergebnisse häufig wenig aussagekräftig. Diesem Problem soll mit einer Benchmark-Suite begegnet werden, die in diesem Beitrag vorgestellt wird. Der PRIOmark verwendet ein neuartiges Lastmodell für die Definition des Zugriffs auf den Sekundärspeicher, das ebenfalls in diesem Beitrag vorgestellt wird.

1 Einleitung

Die CPU-Leistung moderner Hochleistungsrechensysteme steigt stetig an. Entsprechend wächst die Menge bearbeitbarer Daten. Wie eine Untersuchung zeigt, verbessert sich die Effizienz beim Zugriff auf den Massenspeicher nicht in gleichem Maße wie die Prozessorleistung [1]. Es wird deshalb immer schwieriger, die CPU mit genügend Daten für eine optimale Ausnutzung der Rechenleistung zu versorgen. Entsprechend sind Optimierungen an der Leistung des Sekundärspeichers, der die persistente Datenspeicherung des Systems durchführt, von zunehmender Wichtigkeit.

In Hochleistungsrechensystemen werden spezielle Schnittstellen zum Zugriff auf den Sekundärspeicher verwendet. MPI-IO als Teil des Message-Passing-Interfaces (MPI-2 Spezifikation [6]) ist eine weit verbreitete und standardisierte Message-Passing-Schnittstelle in verteilten Applikationen. MPI-IO ermöglicht einen optimierten parallelen Zugriff auf Dateisysteme von Cluster-Computern und ist wesentlich komplexer als die POSIX-Schnittstelle (Portable Operating System Interface), die typischerweise in UNIX-Betriebssystemen verwendet wird.

In diesem Beitrag wird ein Lastmodell zur Definition von I/O-Lasten vorgestellt und ein neuartiges Benchmark-System aus verschiedenen Werkzeugen, das mittels des vorgestellten Lastmodells die I/O-Performance von Sekundärspeicher misst. Die vorgestellte Werkzeugsammlung unterstützt

sowohl MPI-IO als auch POSIX-I/O. Aufgrund der hohen Komplexität von MPI-IO sind die präsentierten Konzepte leicht auf weitere weniger komplexe I/O-Schnittstellen übertragbar.

1.1 Stand der Technik

Benchmarks zur Vermessung der I/O-Performance von Sekundärspeicher werden I/O-Benchmarks genannt [5]. I/O-Benchmarks können hinsichtlich ihrer Lastcharakteristik in zwei Kategorien eingeteilt werden. Die meisten I/O-Benchmarks gehören zur Klasse der *Benchmarks mit definiertem Workload*. Sie nutzen statisch definierte Lastcharakteristiken, die vom Benchmark-Entwickler spezifiziert wurden und nur wenig vom Benutzer verändert werden können. Benchmarks dieser Kategorie haben den Vorteil einer einfachen Nutz- und Vergleichbarkeit. Ihr Nachteil ist, dass die Benchmarkergebnisse häufig nutzlos für den Benutzer sind, da sich der Workload des Benchmarks meist stark vom Workload relevanter Applikationen unterscheidet. Selbst wenn ein Benchmark auf einem Computersystem mit seinem spezifischen Lastverhalten gute Ergebnisse erzielt, kann die I/O-Performance einer echten Applikation mit einem anderen I/O-Verhalten auf dem gleichen Rechensystem sehr schlechte Resultate liefern.

Die zweite Klasse bilden *Benchmarks mit konfigurierbarem Workload*, welche frei durch den Benutzer definierbare Lastcharakteristiken verwenden und bei entsprechend realitätsnaher Workload-Konfiguration sehr gute Ergebnisse erzielen können. Allerdings benötigt der Benutzer dafür detailliertes Wissen über das I/O-Verhalten der Applikation, deren I/O-Leistung ermittelt werden soll.

Die meisten existierenden I/O-Benchmarks wie der weithin gebräuchliche *Bonnie* und *Bonnie++*-Benchmark [2] gehören der ersten Klasse, den *Benchmarks mit definiertem Workload* mit den genannten Problemen, an.

IOzone, einer der weithin gebräuchlichsten I/O-Benchmarks im Linux/UNIX-Umfeld, besitzt als einer der wenigen existierenden I/O-Benchmarks die Möglichkeit komplexe Workloads basierend auf Applikations-I/O-Verhalten zu definieren. Dazu sind zwei Profildateien, die jeweils getrennt die Schreib- und die Lesezugriffe des nachzubildenden I/O-Verhaltens beschreiben, zu definieren. Ein Vermischen der Lese- und Schreiboperationen ist nicht möglich. IOzone bietet keine Unterstützung des Benutzers in der Erstellung dieser komplexen Workloads. [7]

Während die genannten Benchmarks POSIX-I/O als Schnittstelle unterstützt, sollen im Folgenden zwei I/O-Benchmarks vorgestellt werden, die MPI-IO verwenden. Beide MPI-IO-Benchmarks besitzen stark eingeschränkte Möglichkeiten der Konfiguration. *NAS BTIO* als Teil der *NAS Parallel Benchmarks* verwendet einen sehr speziellen Workload, bei dem berechnete Matrizen auf den Sekundärspeicher geschrieben werden, anschließend dort reorganisiert und wieder eingelesen werden [12]. Die *Intel MPI-Benchmarks (IMB)*, früher als *Pallas MPI-Benchmarks* bekannt, enthalten einen I/O-Benchmark, der den größten Teil der MPI-IO Zugriffsfunktionen mittels einer festen, synthetischen Lastcharakteristik testet.

Im Rahmen dieses Beitrags wurde nur eine Auswahl an existierenden I/O-Benchmarks vorgestellt. Bereits dieser Ausschnitt, der einige der wichtigsten I/O-Benchmarks darstellt, zeigt das Problem aktueller I/O-Benchmarks: Entweder sie liefern keine relevanten Ergebnisse oder sie unterstützen komplexe Workloads, deren Konfiguration ohne tiefergehendes Wissen über Applikationsverhalten unmöglich ist.

Im Folgenden wird ein Benchmark-System präsentiert, das eine sehr flexible Konfiguration ermöglicht und den Benutzer bei der Erstellung dieser Konfiguration auf verschiedene Art und Weise unterstützt. Beispielsweise ist eine automatische Analyse des I/O-Verhaltens von Applikationen möglich, um einen Benchmark so zu konfigurieren, dass er das Applikationsverhalten möglichst originalgetreu nachstellt. Dieser Benchmark kann eingesetzt werden, um die I/O-Leistung einer Applikation ohne Installation der Applikation auf verschiedenen Rechnersystemen zu ermitteln und das am besten geeignete System herauszufinden. Das Benchmarksystem gehört der Klasse der *Benchmarks*

mit konfigurierbarem Workload an, geht jedoch mit seinen Möglichkeiten der automatisierten applikationsbasierten Workload-Erstellung über vorhandene Benchmark-Systeme hinaus. Im Folgenden wird er deshalb *applikationsbasierter I/O-Benchmark* genannt. [10].

2 I/O-Lastbeschreibung

Zur Spezifikation des I/O-Lastverhaltens einer realen Applikation als Eingabekonfiguration eines I/O-Benchmarks wird im Folgenden ein I/O-Lastmodell vorgestellt, das trotz der hohen Komplexität von I/O eine einfache Workloaddarstellung mit der Möglichkeit einer realistischen Nachbildung des Applikationsworkloads ermöglicht. Dazu werden zuerst sieben Parameter vorgestellt, die I/O-Performance maßgeblich beeinflussen und anhand dieser werden I/O-Workloads als Trajektorien innerhalb des durch die Parameter aufgespannten sieben-dimensionalen Raumes definiert.

2.1 Der I/O-Workload-Raum

I/O-Leistung kann mittels Durchsatz und Latenz beim Zugriff auf den Sekundärspeicher charakterisiert werden. Beide Parameter hängen stark von der Charakteristik der I/O-Zugriffe ab. Dabei stellt sich die Frage, welche konkreten Parameter einen Einfluss auf die I/O-Performance haben. In [5] werden durch Chen und Patterson fünf Parameter definiert, die aufgrund ihrer Orthogonalität einen Raum aufspannen. Jeder Punkt innerhalb des Raumes entspricht einem I/O-Zugriffsverhalten, also einem spezifischen I/O-Workload. Die fünf Parameter von Chen und Patterson mit ihren entsprechenden Wertebereichen sind:

1. *uniqueBytes* – die Anzahl der insgesamt gelesenen oder geschriebenen Bytes, also die Gesamtmenge an transferierten Daten ($B_u := \{x \mid x \in \mathbb{N}^+\}$),
2. *sizeMean* – die durchschnittliche Größe einer I/O-Anforderung ($S_m := \{x \mid x \in \mathbb{N}^+\}$),
3. *readFrac* – der Anteil gelesener Bytes ($F_r := \{x \mid x \in \mathbb{Q} \mid 0 \leq x \leq 1\}$),
4. *seqFrac* – der Anteil von Anforderungen, die sequentiell auf eine vorherige folgen ($F_s := \{x \mid x \in \mathbb{Q} \mid 0 \leq x \leq 1\}$) und
5. *processNum* – die Anzahl an Prozessen, die parallel I/O durchführen ($N_p := \{x \mid x \in \mathbb{N}^+\}$).

Die fünf definierten Dimensionen sind ausreichend, um I/O-Workloads für nicht verteilte I/O-Systeme zu spezifizieren. Um der höheren Komplexität paralleler Systeme bzw. deren Schnittstellen zum Zugriff auf den verteilten Speicher, gerecht zu werden, ist die Definition weiterer Dimensionen notwendig:

6. *blockFrac* – der Anteil von blockierenden I/O-Anforderungen (nicht-blockierendes asynchrones I/O ist insbesondere in Applikationen der Hochleistungsdatenverarbeitung wichtig, da es eine höhere Parallelität in der Applikation ermöglicht; $F_b := \{x \mid x \in \mathbb{Q} \mid 0 \leq x \leq 1\}$) und
7. *syncFrac* – der Anteil von I/O-Anforderungen, der eine Synchronisierung mit anderen Prozessen benötigt (so ermöglicht MPI-IO I/O-Anforderungen verschiedener Prozesse mit einem gemeinsamen Dateizeiger – in einem solchen Fall ist natürlich eine Kommunikation der Prozesse untereinander unumgänglich; $F_y := \{x \mid x \in \mathbb{Q} \mid 0 \leq x \leq 1\}$).

Neben der Ergänzung zweier neuer Parameter ist eine Erweiterung der Semantik des Parameters *processNum* nötig. Während *processNum* in [5] die Anzahl von Prozessen auf einem Rechner definiert, die parallel I/O durchführen, können diese Prozesse nun auf verschiedenen Rechnerknoten ablaufen. Es entspricht damit nicht mehr der Anzahl einzelner I/O-durchführender Applikationen,

sondern allen Prozessen einer Applikation. Wie auch bei dem in [5] beschriebenen I/O-Modell wird davon ausgegangen, dass alle *processNum* Prozesse die gleichen I/O-Workloads besitzen. Dies ist bei parallelen Applikationen kaum eine Einschränkung, da diese typischerweise nach dem Master-Slave-Paradigma implementiert sind, bei dem entweder nur der Master I/O durchführt oder alle Slaves die gleichen I/O-Operationen ausführen. Insbesondere bei der Verwendung von MPI-IO ist dies nötig, denn I/O-Operationen müssen immer durch alle Prozesse eines MPI-Kommunikators durchgeführt werden. Beide Möglichkeiten sind mittels des vorgestellten Modelles abbildbar.

Mittels der definierten Parameter kann der I/O-Workload-Raum $\mathcal{S}_{\mathcal{IO}}$ wie folgt definiert werden:

$$\mathcal{S}_{\mathcal{IO}} = B_u \times S_m \times F_r \times F_s \times N_p \times F_b \times F_y$$

2.2 I/O-Lasten von Applikationen

Die im letzten Abschnitt erläuterten sieben Parameter definieren Anwendungs-I/O-Verhalten auf sehr abstrakte Art und Weise. Das gesamte I/O-Verhalten einer Applikation wird als ein Punkt innerhalb des I/O-Workload-Raumes aufgefasst:

$$\begin{array}{lll} P_i & = & (u, m, r, s, p, b, y) \\ & \text{mit} & u \in B_u, m \in S_m, r \in F_r, s \in F_s, p \in N_p, b \in F_b \text{ und } y \in F_y \\ & \text{gilt} & P_i \in \mathcal{S}_{\mathcal{IO}} \end{array}$$

Viele Anwendungen ändern während des Applikationslaufes ihr I/O-Verhalten. Eine typische Applikation kann beispielsweise aus den folgenden drei Phasen mit völlig unterschiedlichem I/O-Verhalten bestehen:

1. Lesen von sequentiellen Konfigurationsdaten,
2. Lesen von zufällig angeordneten Applikationsdaten,
3. Schreiben der berechneten Ergebnisse.

Statt eine derartige Anwendung als einen Punkt im Workload-Raum zu charakterisieren, wäre eine Folge von drei Punkten innerhalb des I/O-Workload-Raumes, bei der jeder Punkt eine der oben genannten Phasen korrekt abbildet, eine genauere Darstellung. Der Applikations-Workload, der einer zeitlichen Abfolge der drei Punkte entspricht, kann als Trajektorie innerhalb des Workload-Raumes verstanden werden.

Da die Anzahl der Workload-Punkte einer Trajektorie zur Beschreibung eines Applikations-Workloads variiert werden kann, ergeben sich Beschreibungen auf unterschiedlichen Granularitätsebenen. Wird das Verhalten wie in [5] als ein Punkt, also durch nur sieben Werte charakterisiert, erhält man eine Beschreibung, die keine Programmphasen mehr erkennen lässt. Andererseits ist es möglich, das Applikationsverhalten in einer so feinen Granularität zu beschreiben, dass jede I/O-Anforderung auf einen Punkt innerhalb einer Punktsequenz abgebildet wird. Man erhält dann eine sehr exakte Beschreibung des Verhaltens mit dem Nachteil, dass das Lastprofil sehr komplex zu erstellen und zu analysieren ist. Zwischen den beiden beschriebenen Extremen ist jede Granularität denkbar und kann durch den Nutzer abhängig von seinen Anforderungen verwendet werden.

Eine sequentielle Aneinanderreihung von Punkten innerhalb des Workload-Raumes reicht für die Darstellung eines Applikations-Workloads nicht vollständig aus. Für eine allgemeine Darstellung beliebiger I/O-Lastverhalten muss die Möglichkeit von Schleifen gegeben werden, so dass Teile des Workloads oder der gesamte Workload zyklisch wiederholt werden können. Applikationen wie beispielsweise eine Webserver-Applikation haben eine quasi unendliche Laufzeit in der ständig zyklisch ähnliche I/O-Anforderungen durchgeführt werden. Im Folgenden wird der I/O-Workload von Applikationen derart definiert, dass die Nutzung von `LOOP`-Schleifen, also von Schleifen mit einer

im Voraus definierten Anzahl von Schleifendurchläufen möglich ist. `WHILE`-Schleifen, also Schleifen, deren Schleifendurchlaufanzahl zu Beginn der Abarbeitung unbekannt ist, sollen im Rahmen dieses Beitrags nicht betrachtet werden. Damit geht die Möglichkeit der Definition unendlicher Schleifen im Workload verloren. Es ergibt sich aber der Vorteil, dass jeder Benchmark, der einen auf diese Art und Weise definierten I/O-Workload verwendet, eine endliche Laufzeit besitzt. Dem Problem der unendlichen Laufzeit von Benchmarks, das bei *Benchmarks mit definierten Workloads* nicht auftritt, muss bei der Möglichkeit frei definierbarer Lasten begegnet werden. Nicht-terminierende Benchmarks können schließlich keine Ergebnisse liefern. W_i nennt man einen I/O-Workload, wenn gilt:

- Er ist ein Arbeitspunkt im I/O-Workload-Raum:
  $W_i = P_x$ mit $P_x \in \mathcal{S}_{\mathcal{IO}}$
- Er ist eine Trajektorie im I/O-Workload-Raum:
  $W_i = (P_1, P_2, \ldots, P_i)$ mit $\forall P_x$ mit $1 \leq x \leq i : P_x \in \mathcal{S}_{\mathcal{IO}}$
- Er ist eine sich endlich oft wiederholende Trajektorie im I/O-Workload-Raum:
  $W_i = LOOP(W_j, x)$ mit W_j ist ein I/O-Workload und $x \in \mathbb{N}^+$

Eine derartige I/O-Workload-Beschreibung kann als Dokumentation des Applikations-I/O-Verhaltens oder als Beschreibung für ein Lastgenerations-Werkzeug bzw. als Konfiguration für einen I/O-Benchmark genutzt werden. Der PRIOmark I/O-Benchmark, der in diesem Beitrag vorgestellt wird, verwendet eine derartige I/O-Lastbeschreibung um I/O-Applikationsverhalten nachbilden zu können.

Der Vorgang der Abbildung des I/O-Profils einer Applikation auf eine I/O-Workload-Beschreibung, sowie der umgekehrte Vorgang der Abbildung einer I/O-Workload-Beschreibung auf eine tatsächliche I/O-Last soll im nächsten Abschnitt im Rahmen der für die Abbildungen verantwortlichen Werkzeuge beschrieben werden.

3 Die PRIOmark-Werkzeugsammlung

Wie in Abschnitt 1.1 beschrieben, haben *I/O-Benchmarks mit konfigurierbarem Workload* häufig das Problem, dass sie den Benutzer nicht in der Erstellung komplexer Workload-Konfigurationen unterstützen. Die PRIOmark-Werkzeugsammlung besteht neben dem eigentlichen I/O-Benchmark aus Werkzeugen zur Erstellung komplexer Benchmark-Konfigurationen, die dem I/O-Verhalten von Applikationen entsprechen und außerdem aus Werkzeugen, die den Benutzer in der Analyse der Benchmark-Ergebnisse des PRIOmark unterstützen.

Abbildung 1 zeigt die Architektur der PRIOmark-Werkzeugsammlung. Sie besteht aus fünf Werkzeugen. Sie ermöglichen dem Benutzer die Definition von Workloads mittels zwei unterschiedlicher Wege. Wenn der Benutzer die Applikation sehr gut kennt, kann er I/O-Workloads mittels eines benutzerfreundlichen graphischen Programms, dem Workload-Definition-Tool definieren. Dieses Werkzeug erzeugt ein Workload-Definition-File, das vom I/O-Benchmark verwendet wird, um eine vom Nutzer definierte I/O-Last zu generieren. Das Workload-Definition-File verwendet eine XML-Syntax, um I/O-Workloads mittels des in Absatz 2 vorgestellten Lastmodells zu beschreiben. Ein anderer Weg der Lastdefinition für den Benchmark ist die Benutzung eines I/O-Profilers, der das I/O-Verhalten einer echten Applikation während des Applikationslaufes mitschreibt. Dieses entstehende I/O-Profil kann mittels des Profile-Analyzers in ein Workload-Definition-File konvertiert werden. Dieser Schritt der Profil-Konvertierung ist wichtig, da eine Workload-Definition unterschiedliche Abstraktionslevel besitzen kann, wie es in 2.2 beschrieben ist. Ein I/O-Profil hingegen hat immer die feinste Granularität der I/O-Beschreibung, da jede I/O-Anforderung protokolliert

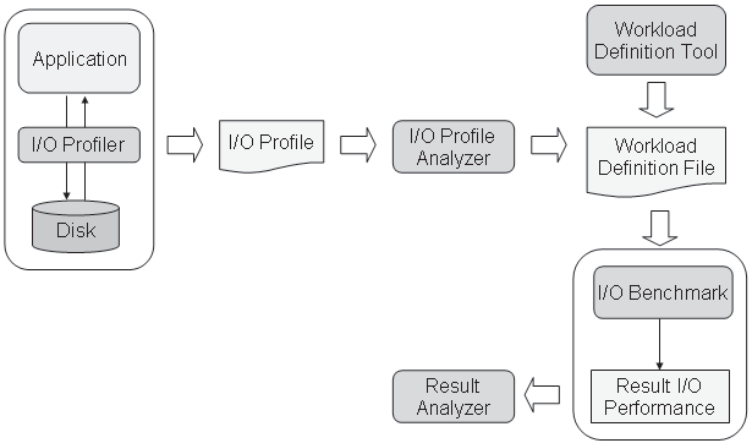


Abbildung 1: Architektur der PRIOmark Werkzeugsammlung

wird. Schließlich gibt es einen grafischen Result-Analyzer, der den Benutzer beim Verstehen, der zum Teil sehr komplexen Messergebnisse mittels verschiedener Analysemethoden unterstützt.

Anschließend werden die fünf Werkzeuge der PRIOmark Werkzeugsammlung etwas detaillierter vorgestellt.

3.1 Workload-Definition-Tool

Dieses Werkzeug ermöglicht die Definition von I/O-Workloads in einem grafischen Frontend. Workloads werden als einfache State-Charts definiert, wobei jeder Zustand einem Punkt im I/O-Workload-Raum entspricht und mit den entsprechenden sieben Dimensionen definiert wird. Zyklen innerhalb der Graphen sind erlaubt und entsprechen den in der formalen Definition beschriebenen `LOOP`s. An der rückführenden Kante der Zyklen wird die Anzahl der Schleifendurchläufe angegeben. Abbildung 2 zeigt einen Screenshot des Workload-Defintion-Tools auf der rechten Seite. Auf der linken Seite ist der Result-Analyzer abgebildet, der später erklärt wird.

3.2 I/O-Profiler

Der I/O-Profiler ist ein Werkzeug zum Protokollieren der I/O-Anforderungen einer spezifischen Applikation in eine Datei, die I/O-Profil genannt wird. Damit entspricht ein derartiges Profil sehr exakt dem I/O-Verhalten einer Applikation. Am Lehrstuhl für Rechnerarchitektur der Uni Rostock wurden zwei Profiler entwickelt, die sowohl die POSIX-I/O-Schnittstelle als auch die MPI-IO-Schnittstelle protokollieren.

Ein Ziel während der Entwicklung der Tools war die Erstellung von I/O-Profilen selbst von Applikationen, die nur in kompilierter Form vorliegen. Deshalb wurden die Profiler als Shared-Libraries für Linux-Systeme implementiert, die ein dynamisches Linken des Profilers gegen die Applikation ermöglichen und die I/O-Aufrufe abfangen, um sie mitzuprotokollieren. Die originalen I/O-Aufrufe werden anschließend aus der Wrapper-Funktion aufgerufen, um eine für die Applikation transparente Protokollierung zu ermöglichen.

Diese Architektur ermöglicht es, einen verfälschenden Einfluss des Profilers auf das zu vermessende Softwaresystem nahezu auszuschließen. Der Profiler ist aus Sicht der Applikation völlig transparent, so dass diese ihre Aufgaben mit und ohne I/O-Profiler identisch durchführt. Dies führt dazu, dass auch das aufgezeichnete I/O-Profil identisch zu einem regulären Applikationslauf ist. Wie Messungen ergaben, ist selbst der Performanceeinfluss des Profilers auf den Applikationslauf

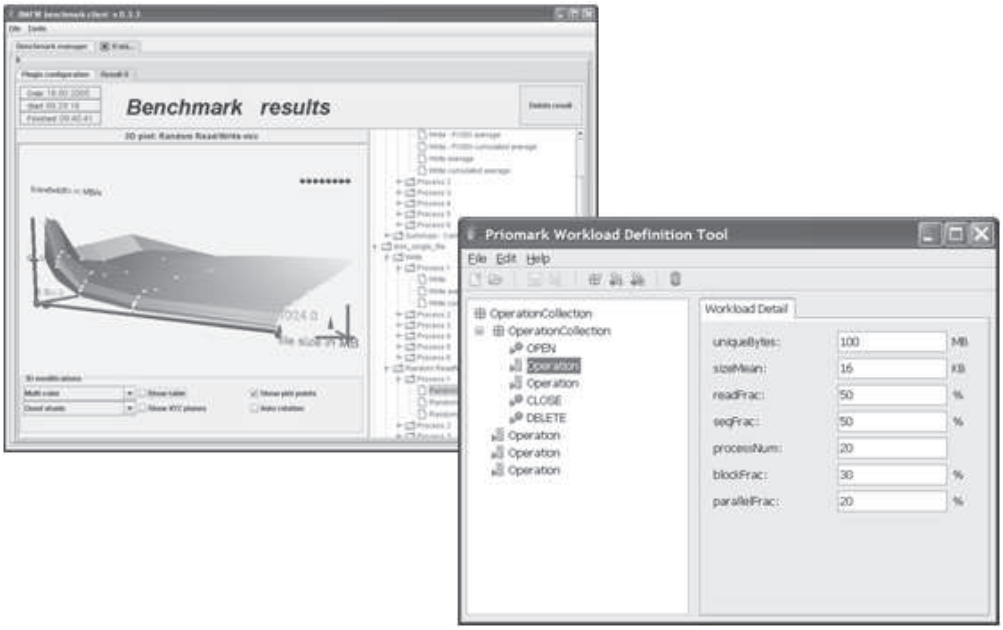


Abbildung 2: Screenshots des PRIOmark-Result-Analyzers und des Workload-Definition-Tools

während der Aufzeichnung gering [1]: In einer Applikation, die einen synthetischen Workload verwendete, der neben wenigen Initialisierungen ausschließlich aus I/O-Anforderungen bestand, war der Performanceverlust geringer als 10 Prozent. In einer realen Applikation mit weitaus weniger I/O-Aufrufen, wird er deshalb noch geringer.

3.3 I/O-Profile-Analyzer

Die vom I/O-Profiler generierten I/O-Profile können nicht direkt vom I/O-Benchmark verarbeitet werden. Sie müssen in einem Zwischenschritt in ein Workload-Definition-File konvertiert werden. Diese Aufgabe, die nicht ohne Interaktion mit dem Benutzer erfolgen kann, wird von dem grafischen Werkzeug I/O-Profile-Analyzer übernommen. Die Nutzerinteraktionen sind nötig, um die Granularität der Zielbeschreibung bestimmen zu können. Die aktuelle Implementierung des Profile-Analyzers ermöglicht nur eine direkte Abbildung des Profiles auf eine Workload-Beschreibung, d. h. jede I/O-Anforderung wird auf einen Punkt im I/O-Workload-Raum abgebildet. Zukünftig wird der Benutzer das Zusammenfassen dieser Punkte mittels Bedingungen steuern können, um so grobgranulare Lastbeschreibungen zu erhalten. Der Benutzer wird beliebige Bedingungen formulieren können, die sich auf I/O-Anfragen und deren Parameter beziehen. Da das I/O-Profil auch die genaue zeitliche Abfolge von Requests enthält, sind so auch Bedingungen über die Zeit möglich. Constraint-Definitionen die das I/O-Profil der in Abschnitt 2.2 eingeführten Beispielapplikation auf drei Punkte reduzieren, die den genannten drei Phasen entsprechen, können beispielsweise sein:

- Zusammenfassung aller I/O-Anforderungen, die vor dem Schließen der Datei */etc/prog.conf* stattfinden,
- Zusammenfassung aller darauf folgenden Anforderungen bis zum ersten Schreibvorgang,
- alle restlichen I/O-Requests entsprechen dem dritten Workload-Punkt.

3.4 I/O-Benchmark

Das zentrale Element der PRIOmark Werkzeug-Sammlung zur eigentlichen Messung der I/O-Performance ist der PRIOmark I/O-Benchmark. Er verwendet die Workload-Beschreibung, die dem I/O-Workload einer Applikation entspricht, um diese so originalgetreu wie möglich nachzubilden.

Um einen Arbeitspunkt in der Workload-Beschreibung möglichst exakt durch einen realen Workload darzustellen, wird jeder Arbeitspunkt auf eine Sequenz von I/O-Operationen nach den folgenden Regeln abgebildet:

- Die Sequenz besteht aus $n = \left\lfloor \frac{uniqueBytes}{sizeMean \cdot processNum} \right\rfloor$ I/O-Anforderungen. Da n ganzzahlig sein muss, ist der entsprechende Wert zu runden. Für eine gute Approximation muss der Wert für *uniqueBytes* sehr viel größer sein als $sizeMean \cdot processNum$.
- Die n I/O-Anforderungen werden abhängig von den Werten der drei Dimensionen *readFrac*, *blockFrac* und *syncFrac* auf verschiedene I/O-Funktionen abgebildet. Wenn beispielsweise *readFrac* den Wert 0.5 hat, besteht die Hälfte der Anforderungen aus `read`-Operationen, von denen der mit *blockFrac* und *syncFrac* angegebene Anteil blockierend und synchronisiert ist.
- Die ermittelten I/O-Anforderungen werden mit zufälligen Daten ausgeführt. Dabei wird der Anteil *seqFrac* an Anforderungen sequentiell ausgeführt, während der Rest zufällig ermittelte Positionswerte zum Datenzugriff innerhalb der Datei verwendet. Die Größe der Anforderungen entspricht dem mittels *sizeMean* angegebenen Wert.

Viele Lastgeneratoren variieren einzelne Lastparameter wie `sizeMean` mittels verschiedener statistischer Verteilungen. Pro Parameter wird eine Verteilung definiert und es müssen statistische Werte zur konkreten Definition der verwendeten Verteilung angegeben werden. Bei diesem Modell wurde bewusst auf ein solches Verfahren verzichtet, da der ohnehin schon 7-dimensionale I/O-Workload-Raum durch ein solches Vorgehen weiter erhöht werden würde. Selbst wenn man für alle variierbaren Parameter eine Normalverteilung festlegt, kommt pro variierbarem Parameter die entsprechende Standardabweichung σ als weitere Dimension hinzu. Vielmehr ist es mittels des vorgestellten Modelles möglich, beliebige Verteilungen mittels einer Trajektorie aus mehreren Punkten im Workload-Raum zu definieren. Aus diesen Gründen werden Parameter, wie die durchschnittliche I/O-Anforderungsgröße durch den PRIOmark nicht statistisch variiert. Wie die im Anschluss vorgeführten Messungen anhand einer Beispielapplikation zeigen, sind die so erzielten Ergebnisse trotz dieser Vereinfachung hinreichend genau.

Pfad und Dateiname der Ausgabedatei werden durch den Benchmarknutzer ebenfalls bestimmt. Da Dateisysteme auf Verzeichnispfade abgebildet werden, kann auf diese Art und Weise ein Vergleich von Dateisystemen bei Verwendung des gleichen Workloads stattfinden. Abschnitt 4 zeigt einen derartigen Vergleich zwischen den Dateisystemen NFS und PVFS.

Während der Ausführung der I/O-Anforderungen misst der Benchmark die Ausführungszeiten der Operationen. Die I/O-Arbeitspunkte der Workload-Definition werden dabei in der spezifizierten Reihenfolge ohne Pausen abgearbeitet. Entsprechend gibt es an dieser Stelle Unterschiede zum originalen Workload, in dem durch die Berechnungen der Applikation Pausen zwischen den I/O-Anforderungen auftreten. Die Frequenz von Anforderungen und damit die Belastung des I/O-Systems insgesamt ist also während der Wiedergabe des Profils normalerweise höher als im originalen Fall. Dies lässt eine geringere I/O-Performance bei der Profilwiedergabe gegenüber dem Original vermuten. Wie stark sich die Unterschiede zwischen Messung und Original in der Realität sind, zeigen die im Abschnitt 4 präsentierten Messungen. Nicht zu vermeiden ist, dass Applikationen auf unterschiedlichen Plattformen verschiedene Profile aufgrund von plattformabhängigen Optimierungen des Applikationsentwicklers besitzen könnten. Dies würde zu unter Umständen signifikant unterschiedlichen Ergebnissen bei der Wiedergabe des I/O-Profils auf anderen als der Aufzeichnungsplattform führen. Während dies bei der Verwendung von POSIX-I/O tatsächlich

passieren kann, ist es bei der Nutzung von MPI-IO auszuschließen, da die MPI-IO-Softwareschicht völlig von der Hardware abstrahiert und selbst die Optimierungen abhängig von der verwendeten Hardware durchführt. Plattformabhängige Codes seitens des Entwicklers sollten daher auf dieser Ebene vermieden werden.

Der Benchmark speichert für alle unterschiedlichen Zugriffsarten die entsprechenden Leistungswerte ab. Dies erlaubt eine sehr detaillierte Auswertung der Ergebnisse bei der der PRIOmark-Result-Analyzer den Benutzer unterstützen kann. Dieses Werkzeug wird im Folgenden kurz vorgestellt.

3.5 Result-Analyzer

Der PRIOmark-Result-Analyzer ist ein in Java entwickeltes Werkzeug, das den Benutzer dabei unterstützt, die komplexen Ausgaben des PRIOmark I/O-Benchmarks zu analysieren. Dafür stehen eine große Anzahl an Analysemöglichkeiten zur Verfügung. Unter anderem ermöglicht der Result-Analyzer die Erstellung von zwei- oder dreidimensionalen Graphen zur Analyse von Messwerten, die von unterschiedlich vielen Variablen abhängen. Diese Graphen können gedreht werden oder es kann in sie hineingezoomt werden, um auf diese Art und Weise die Zusammenhänge visuell zu beurteilen.

Weiterhin ermöglicht der PRIOmark-Result-Analyzer eine entfernte Ausführung des PRIOmark I/O-Benchmarks über eine Netzwerkverbindung, um die Ausführung auf entfernten Rechnern wie bspw. Hochleistungsrechnern, die nur über Netzwerk erreichbar sind, steuern zu können.

4 Beispielmessungen

Ziel dieses Abschnitts ist es, zu zeigen, dass die Messungen, die mit dem PRIOmark Benchmark ermittelt werden, wirklich relevant für den Benutzer sind. Dazu wurden zwei Messungen durchgeführt, deren Ergebnisse hier präsentiert werden. Die erste Messung vergleicht die die I/O-Performance einer Applikation mit der Performance, die vom PRIOmark I/O-Benchmark durch Nachahmung des Applikations-I/O-Verhaltens gemessen wurde, während die zweite Messung die I/O-Performance unter unterschiedlichen Lasten miteinander vergleicht.

4.1 Vergleich von Applikations-I/O-Leistung mit PRIOmark-Messwerten

Klassische Anwendungen sehen keine Möglichkeit vor, ihre I/O-Performance zu ermitteln. Die Ermittlung dieser Leistungswerte ist aber wichtig, um die Genauigkeit der Appoximation des Applikations-I/O-Verhaltens durch den PRIOmark in Hinblick auf die originale Leistung zu bewerten. Daher dient im Rahmen dieser Messungen ein MPI-IO-Benchmark als Applikation, der selbst seine I/O-Performance bestimmen kann. Die vom klassischen Benchmark ermittelte Leistung wird dann mit dem Ergebnis des PRIOmark I/O-Benchmarks verglichen, der das I/O-Verhalten des klassischen I/O-Benchmarks nachbildet.

Die Ergebnisse müssen im Original und in der Nachbildung ähnliche Werte erreichen, wenn das beschriebene Konzept eines I/O-Benchmarks zur Nachbildung von Applikations-I/O-Verhalten funktioniert. In der Praxis kann schließlich jede Applikation anstatt des klassischen Benchmarks verwendet werden, um deren I/O-Leistung auf spezifischen Plattformen zu bewerten. Die Messungen wurden auf den beiden Dateisystemen Network File System (NFS) [11], das noch immer in vielen netzbasierten Umgebungen verbreitet ist und auf dem Parallel Virtual File System (PVFS) [4] durchgeführt. PVFS wird auf Hochleistungsrechensystemen verwendet und ist optimiert für den Zugriff auf parallele Dateisysteme.

Die in den Messungen verwendete Applikation ist der Strided-Benchmark[9][8]. Da der Strided-Benchmark ein reiner MPI-IO-Benchmark ist, werden die Messergebnisse nicht durch andere MPI-

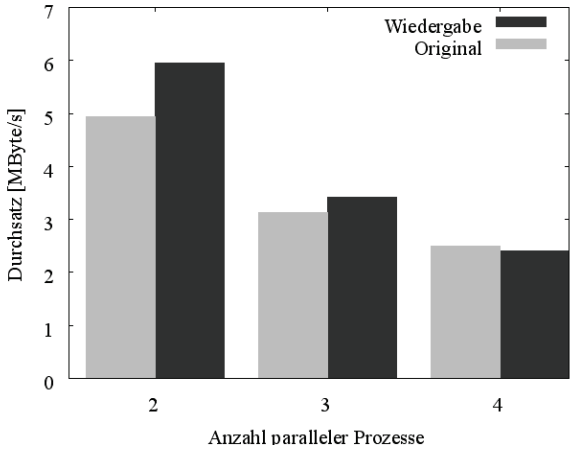


Abbildung 3: NFS-Leistung

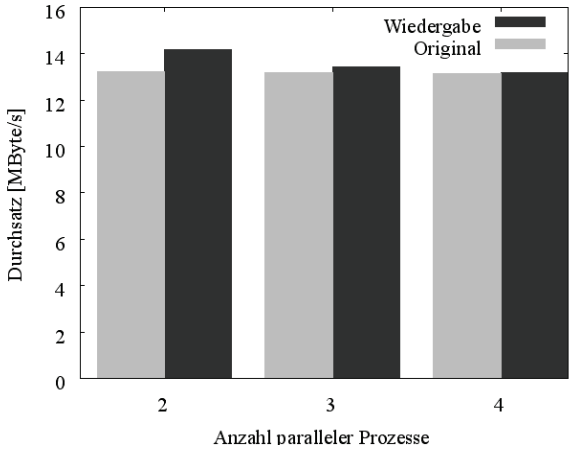


Abbildung 4: PVFS-Leistung

oder POSIX-Aufrufe beeinflusst. Die Messumgebung ist ein PC-Cluster bestehend aus 8 Knoten mit einem Linux-Betriebssystem und der MPICH2-Implementierung für MPI. Drei der 8 Knoten haben jeweils 2 Xeon-Prozessoren (2.8 GHz) mit 2 GByte Hauptspeicher und die restlichen Knoten verwenden 2 Pentium III-Prozessoren (1.4 GHz) mit ebenfalls 2 GByte Hauptspeicher. Für die Messungen wurden maximal 4 der Knoten verwendet.

Abbildung 3 zeigt die Ergebnisse der Messungen auf einem NFS-Dateisystem. Die hellen Balken zeigen die erreichte Bandbreite pro Prozess in Abhängigkeit von der Prozessanzahl, die durch den originalen Strided-Benchmark erreicht wurden. Die dunkleren Balken zeigen die erreichten Bandbreiten des PRIOmark-Benchmarks, der die I/O-Last des Strided simuliert. Dabei ist deutlich erkennbar, dass beide Messungen ein ähnliches Verhalten in Abhängigkeit von der Prozessanzahl aufweisen. Außerdem wird deutlich, dass NFS mit steigender Prozessanzahl die zur Verfügung stehende Bandbreite auf die Prozesse aufteilt, so dass die Bandbreite pro Prozess sinkt. Bei PVFS hingegen, dessen Messungen in Abbildung 4 dargestellt werden, bleibt die Bandbreite pro Prozess mit steigender Prozessanzahl etwa konstant. PVFS skaliert deutlich besser mit der Prozessanzahl. Begründet ist dies in der Anzahl der Knoten, die die Daten speichern. Während es bei NFS lediglich einen Server gibt, der nur die Netzwerkbandbreite für alle Prozesse zur Verfügung stellen kann, gibt es bei PVFS mehrere I/O-Knoten. Dies ermöglicht einen parallelen Zugriff mehrerer Prozesse mit voller Bandbreite auf Daten, die auf unterschiedlichen Knoten gespeichert werden. Beim PVFS-Dateisystem ist der Unterschied zwischen originaler und wiedergegebener Bandbreite sogar geringer als bei NFS. Es zeigt sich, dass die mittels des PRIOmark gemessene Bandbreite häufig höher ist als die Originalbandbreite. Dies spricht gegen die genannte Vermutung, dass die mit dem PRIOmark ermittelte Performance aufgrund der höheren Belastung des I/O-Systems durch die fehlenden Pausen geringer ist. Grund dafür sind die leicht unterschiedlichen Messmethoden des Strided und des PRIOmark. Der Strided-Benchmark misst die komplette Laufzeit des Benchmarks inklusive allen Operationen, die zwischen den I/O-Operationen ausgeführt werden. Der PRIOmark hingegen misst die Ausführungszeiten der I/O-Aufrufe ohne Overhead und ermittelt damit etwas höhere Bandbreiten.

Der verwendete Strided-Benchmark nutzt als I/O-Zugriffsverhalten einen synthetischen I/O-Workload und entspricht damit keiner realitätsnahen Applikation. Dennoch zeigen die Messungen, dass die Nachbildung dieses synthetischen Verhaltens gut gelingt. Der Vergleich der I/O-Performance echter Anwendungen mit den PRIOmark-Ergebnissen ist nicht ohne Probleme möglich, da Applikationen typischerweise ihre I/O-Performance selbst nicht ermitteln. Mittels Instrumentierung des Applikationscodes wäre die Messung möglich. In zukünftigen Entwicklungen werden

Untersuchungen diesbezüglich angestellt. Im Folgenden werden erste Messwerte präsentiert, die belegen, dass der PRIOmark-I/O-Benchmark bei Verwendung von vier Lasten realer Applikationen auf der gleichen Plattform signifikant unterschiedliche Ergebnisse erzielt. Dies unterstreicht die Notwendigkeit eines Benchmarksystems wie das präsentierte.

4.2 Lastabhängige I/O-Performance

Um zeigen zu können, dass er die Konfiguration unterschiedlicher Lastszenarien, wie sie in der Realität vorkommen unterschiedliche Auswirkungen auf die I/O-Performance haben, wurden weitere Messungen durchgeführt, die vier typische Lastverhalten miteinander vergleichen (Workstation, Online-Transaction-Processing-Applikation – also eine Datenbank mit Transaktionen, Webserver, Fileserver). Die I/O-Charakteristiken wurden dem Beitrag [3] entnommen und mittels des Workload-Definition-Tools wurde eine Konfiguration für den PRIOmark I/O-Benchmark erstellt. Abbildung 5 zeigt die Ergebnisse dieser Messungen. Während der klassische Workstation-Workload,

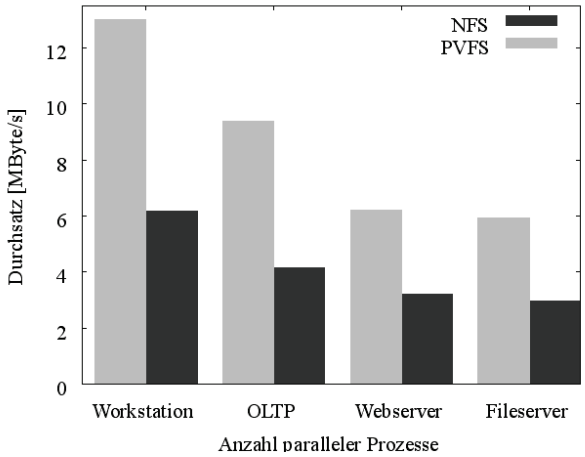


Abbildung 5: Vergleich der I/O-Performance bei unterschiedlichen Lasten

wie er auf Heimcomputern verwendet wird, die beste Performance der vier vermessenen Lastverhalten erzielt, erreicht das System bei Verwendung der Fileserver-Last weniger als die Hälfte der Performance der Workstation. Dies liegt zum anderen an der höheren Sequentialität der Daten bei der Workstation und zum anderen an den weniger stark differierenden Anforderungsgrößen. Deutlich ist auch bei diesen Messungen der Performance-Unterschied zwischen PVFS und NFS. Trotz dessen, dass nur ein PVFS-I/O-Node verwendet wurde, ist die PVFS-Performance ca. doppelt so hoch wie die NFS-Performance bei gleichem Lastszenario.

Insgesamt zeigen die Messungen, dass der präsentierte Ansatz der Messung von I/O-Leistung anhand von vorher mitgeschriebenen I/O-Verhalten eine mögliche Lösung ist, um einfach benutzerrelevante I/O-Leistungsdaten zu ermitteln.

5 Zusammenfassung

Dieser Beitrag präsentiert einen Ansatz, um eines der größten Probleme aktuell verfügbarer I/O-Benchmarks zu lösen. Verfügbare I/O-Benchmarks messen Leistungswerte anhand von synthetischen Lastcharakteristiken mit geringem Bezug zu realistischen Lasten. Dieser Beitrag stellt ein Benchmark-System vor, das Applikations-I/O-Verhalten protokolliert und in einem abstrakten Darstellungsformat dem Benutzer zur Verfügung stellt. Der Benchmark verwendet die abstrakte Ver-

haltensbeschreibung und nutzt damit das I/O-Verhalten dedizierter Applikationen, um realistische Leistungswerte zu ermitteln. In diesem Beitrag wird das abstrakte Workload-Modell zur Beschreibung des I/O-Verhaltens detailliert beschrieben und anhand von Messungen gezeigt, dass mittels der präsentierten Beschreibung das Applikationsverhalten hinreichend genau wiedergegeben werden kann, um applikationsorientierte und aufgabengetreue Leistungswerte zu ermitteln.

Literatur

[1] Ralf Behnke. Konzeption und Implementierung eines Profilers für MPI-IO in verteilten HPC-Anwendungen. Master's thesis, Universität Rostock, 2006.

[2] Bonnie Benchmark. IOzone Filesystem Benchmark. http://www.textuality.com/bonnie, 2005.

[3] Ray Bryant, Dave Raddatz, and Roger Sunshine. Penguinometer: A new file-i/o benchmark for linux. In *Proceedings of the 5th Annual Linux Showcase and Conference*, Oakland, CA, 2001.

[4] Philip H. Carns, Walter B. Ligon III, Robert B. Ross, and Rajeev Thakur. PVFS: A parallel file system for linux clusters. In *Proceedings of the 4th Annual Linux Showcase and Conference*, pages 317–327, Atlanta, GA, 2000. USENIX Association.

[5] Peter M. Chen and David A. Patterson. A New Approach to I/O Performance Evaluation – Self-Scaling I/O Benchmarks, Predicted I/O Performance. *ACM Transactions on Computer Systems*, 12(4):308–339, 1994.

[6] William Gropp, Steven Huss-Lederman, Andrew Lumsdaine, Ewing Lusk, Bill Nitzberg, William Saphir, and Marc Snir. *MPI — The Complete Reference: Volume 2, the MPI-2 Extensions.* MIT Press, 1998.

[7] IOzone Filesystem Benchmark. IOzone Filesystem Benchmark. http://www.iozone.org/, 2005.

[8] Michael Krietemeyer, Heiko Kopp, Daniel Versick, and Djamshid Tavangarian. Environment for I/O Performance Measurement of Distributed and Local Secondary Storage. In *Proceedings of the International Conference on Parallel Processing Workshops*, pages 501–508. IEEE, 2005.

[9] Michael Krietemeyer, Daniel Versick, and Djamshid Tavangarian. The PRIOmark – Parallel I/O Benchmark. In *Proceedings of the IASTED International Conference on Parallel and Distributed Computing and Networks*, pages 595 – 600, 2005.

[10] Michael Krietemeyer, Daniel Versick, and Djamshid Tavangarian. Workload-basierte Klassifikation von Benchmarks für lokale und verteilte I/O-Systeme. In Klaus Heidtmann Bernd Wolfinger, editor, *Leistungs-, Zuverlässigkeits- und Verlässlichkeitsbewertung von Kommunikationsnetzen und verteilten Systemen*, pages 119–127. Universität Hamburg, Fachbereich Informatik, 2005.

[11] Brian Pawlowski, Chet Juszczak, Peter Staubach, Carl Smith, Diane Lebel, and Dave Hitz. NFS version 3: Design and implementation. In *USENIX Summer*, pages 137–152, 1994.

[12] Parkson Wong and Rob Van der Wijngaart. NAS Parallel Benchmarks I/O Version 2.4. Technical report, 2003.

Rates of convergence for closed cycles of Infinite Server Queues

Kersten Tippner
Hamburg University, Department of Mathematics
kersten_eckert@gmx.net

Abstract

For cyclic networks of Infinite Server (IS) queues we derive bounds for the distance of the system in its transient phase from equilibrium. The metric used to measure this distance is the total variation of probability distributions. We are able to prove that the distance between some transient state and the equilibrium for the general network under this metric can be bounded by the distances between certain transient states and the equilibria of several much more simple networks. In certain cases, we derive explicit bounds for these simple networks, partly found in literature on related systems in the realm of elementary random walks.

Introduction

Queueing networks are versatile models, e.g. for telecommunication systems, computer networks, production systems or population dynamics. Performance evaluation for these networks often relies on having explicitly the steady state probabilities for the network states at hand.
The standard class of models which satisfies this property is the class of separable or product form networks, which was popularized in computer science by Kleinrock[1] and later on by the celebrated BCMP[2] networks.
BCMP networks encompass nodes with service disciplines processor sharing, last-come-first-served (preemptive resume), and infinite servers, and exponential first-come-first-served nodes.
Although the steady state probabilities for the network states are explicitly known, many quantities that are of interest for assessing quality of service (QoS) in such models are up to now not accessible by analytical methods, especially customer-oriented performance measures as, for instance, passage times or waiting times.
Here we have to resolve the problem of assessing QoS by simulation. One of the hard problems connected with steady state simulation of large systems is to decide whether an already long run of the simulation has reached steady state.

The dynamics of BCMP networks are described by Markov processes in continuous time - which turns our problem into the direction of quantifying the distance (in some more or less precise meaning) of the Markov process from its steady state. This is at present one of the most challenging problems in simulating the steady state of large systems, which is often done by Markov Chain Monte Carlo (MCMC) methods.

In this paper we investigate a subclass of BCMP networks: We assume to have only Infinite Server nodes (IS) and we restrict our investigation to closed cycles of Infinite Server queues. We further consider the evolution of the networks on a discrete time scale. The reason for these

[1] [Klei75]
[2] [BCMP75]

restrictions are twofold: Firstly, we try to develop further the theory of discrete-time queueing networks, which have become ever more popular with the advent of ATM as the protocol of high speed networks, and secondly, we want to start with rather simplified models, where results from literature are of value for our investigations.

Notation: For the set $\{1,\dots,N\}$ we denote by $n\oplus 1 = n+1$ for $1\le n\le N-1$ and $N\oplus 1 = 1$, and by $n\ominus 1 = n-1$ for $2\le n\le N$ and $1\ominus 1 = N$.

1 Closed Cycles of Infinite Server Queues

We consider a closed cycle of N Infinite Server (IS) queues in discrete time with K distinguishable customers circulating. Here IS means that at all nodes there is ample service capacity. We further assume that
• Each of the K customers is assigned some type $\vartheta\in\{1,\dots,T\}$.
• If at time t customer i (of type $\vartheta(i)$) is present (and thus being served) at node $\mathfrak{j}(i)$, he will leave this node with probability $p_{\mathfrak{j}(i),\vartheta(i)}$, independent of anything else, with probability $1-p_{\mathfrak{j}(i),\vartheta(i)}$ this customer will stay-on for at least one further service quantum.
• A customer leaving node $\mathfrak{j}$, $\mathfrak{j}=1,\dots,N$, is transferred to node $\mathfrak{j}\oplus 1$.

Keeping track of every customer's type and their actual node position allows a Markov chain description of the cycle's evolution. We therefore observe system states

$$\hat{\mathfrak{e}} = ((\mathfrak{j}(1),\vartheta(1)),(\mathfrak{j}(2),\vartheta(2)),\dots,(\mathfrak{j}(K),\vartheta(K))),\quad \mathfrak{j}(k)\in\{1,\dots N\},\vartheta(k)\in\{1,\dots,T\},\qquad (1.1)$$

Here $\mathfrak{j}(i)$ denotes the position of the i-th customer, $i=1,\dots,K$, and $\vartheta(i)$ his type.[3]
Let $\hat{E}$ be the state space consisting of the above states.

Theorem 1.1. In a closed cycle of Infinite Server Queues with the above characteristics, the equilibrium distribution for the network process is given by

$$\hat{\pi}(\hat{\mathfrak{e}}) = G(K,N)^{-1}\prod_{i=1}^{K} p_{\mathfrak{j}(i),\vartheta(i)}^{-1},\qquad (1.2)$$

$G(K,N)^{-1}$ being the norming constant given by

$$G(K,N) = \sum_{\mathfrak{j}(1)=1}^{N}\cdots\sum_{\mathfrak{j}(K)=1}^{N}\prod_{i=1}^{K} p_{\mathfrak{j}(i),\vartheta(i)}^{-1}.\qquad (1.3)$$

Proof. Closed cycles of Infinite Server queues are special cases of the networks examined in [Hen90], [Hen91] and [Miya94], for instance. The proof for the above theorem in a wider context can be found in [Hen90] (Th. 1) and [Eck03] (Th. 2.1). □

2 Speed of convergence for closed cycles of queues

In performance analysis, it is well-known that the selection of appropriate metrics is sometimes a difficult task, which, on the other hand, may be just the key to successfully assessing the systems' behaviour. Usually the speed of convergence is measured by means of the distance between

[3] Although the customer type does not change over time, we record it, because we can later on optimize with respect to initial type selection and grouping customers in batches with stochastically identical behaviour.

the actual transient state and the equilibrium state. For our purpose the TOTAL VARIATION DISTANCE between probability distributions came out as a well-tailored metric.
In fact, in our cycles with distinguishable customers, it permits to reduce to compute, or at least to bound the speed of convergence for a cycle with K customers to K systems with 1 customer each. We shall prove this and then provide examples, partly from the literature, where the 1-customer system can be dealt with effectively.

2.1 The total variation distance

Definition 2.1. *Total variation*
The total variation between two distributions Q_1, Q_2 on a state space E is given by

$$\|Q_1 - Q_2\|_{TV} := \max_{A \subset E} |Q_1(A) - Q_2(A)| = \frac{1}{2} \sum_{\mathfrak{e} \in E} |Q_1(\mathfrak{e}) - Q_2(\mathfrak{e})|. \tag{2.1}$$

Consider now a homogeneous Markov chain $X = (X_n : n \in \mathbb{N})$ on finite state space E, irreducible and aperiodic, and with transition matrix $P = (P(\mathfrak{e}, \mathfrak{e}') : \mathfrak{e}, \mathfrak{e}' \in E)$ and stationary distribution $\pi = (\pi(\mathfrak{e}) : \mathfrak{e} \in E)$.
For $t \geq 1$, we denote by P^t the t-step transition matrix of X. Then $P^t(\mathfrak{e}_0, \bullet) := (P^t(\mathfrak{e}_0, \mathfrak{e}), \mathfrak{e} \in E) = (P(X_t = \mathfrak{e} \mid X_0 = \mathfrak{e}_0), \mathfrak{e} \in E)$, $t \in \mathbb{N}$, is the distribution of X at time t given the process is started in a fixed state $\mathfrak{e}_0$.

Definition 2.2. Assume X is started in $\mathfrak{e}_0$. Then the total variation distance between the transient state at time t and the equilibrium is denoted by

$$d(\mathfrak{e}_0, t) := d(P^t(\mathfrak{e}_0, \bullet), \pi) := \|P^t(\mathfrak{e}_0, \bullet) - \pi\|_{TV} \tag{2.2}$$

The maximal variation between the transient state at time t and the equilibrium is defined by

$$d^*(t) := \max_{\mathfrak{e}_0} d(\mathfrak{e}_0, t) = \max_{\mathfrak{e}_0} \|P^t(\mathfrak{e}_0, \bullet) - \pi(\bullet)\|_{TV}. \tag{2.3}$$

2.2 Bounds for total variation distance in IS cyclic networks

Consider now the cycle from Section 1 with only customer i (of fixed type $\vartheta(i) \in \{1, \ldots, T\}$) circulating. Then the state space of this network is

$$E_i := \{\mathfrak{e}(i) = (\mathfrak{j}(i), \vartheta(i)) : \mathfrak{j}(i) \in \{1, \ldots, N\}\}. \tag{2.4}$$

Then $\hat{E} = \prod_{i=1}^{K} E_i$, i. e. $\hat{E}$ is the Cartesian product of K state spaces, one state space for each customer.
We denote by $\hat{X}$ and X_i Markov chains for the network with all customers, respectively with just customer i circulating; the associated one-step transition kernels are denoted by $\hat{P}$ and P_i, respectively. We want to relate the speed of convergence of $\hat{X}$ to its equilibrium to the the speed of convergence of the $X_i, i = 1, \ldots, K$, to their equilibria.
For this purpose, we set

$$d_{\hat{E}}(\hat{P}^t(\mathfrak{e}_0, \bullet), \hat{\pi}) :=\| \hat{P}^t(\mathfrak{e}_0, \bullet) - \hat{\pi} \|_{TV}, \tag{2.5}$$

and

$$d_{E_i}(P_i^t(\mathfrak{e}_0(i), \bullet), \pi_i) :=\| P_i^t(\mathfrak{e}_0(i), \bullet) - \pi_i \|_{TV}, \tag{2.6}$$

with $\mathfrak{e}_0 = (\mathfrak{e}_0(1), \ldots, \mathfrak{e}_0(K))$ being the initial state of the network process $\hat{X}$ and π_i being the steady-state distribution of X_i, which is just the i-th marginal distribution of $\hat{\pi}$. Our main general bound is now provided by the following theorem.

Theorem 2.3. [TL07] Assume that the network process X is started in initial state $\mathfrak{e}_0 = (\mathfrak{e}_0(1), \ldots, \mathfrak{e}_0(K))$. Then the total variation distance between the transient state of X at time t and its equilibrium (on $\hat{E}$) can be bounded as follows

$$d_{\hat{E}}(\hat{P}^t(\mathfrak{e}_0, \bullet), \hat{\pi}) \leq \sum_{i=1}^{K} d_{E_i}(Pi^t(\mathfrak{e}_0(i), \pi_i) \tag{2.7}$$

Proof. The proof for this theorem will be published in a forthcoming preprint at the Department of Mathematics, University of Hamburg. □

3 Speed of convergence for 1-customer cycles

In the remaining part of the paper we describe prototype examples in which the total variation distance of the simple 1-customer networks can be estimated or bounded by various methods. Our first approach is the construction of so-called strong stationary times, which enable us to measure the speed of convergence of 1-customer networks, introduced in Example 3.1.
As the construction of strong stationary times is complicated even for clockwise moving customers in the cycle, we then try to bound total variation distance
$d_{E_i}(P_i^t(\mathfrak{e}_0(i), \bullet), \pi_i) = \| P_i^t(\mathfrak{e}_0(i), \bullet) - \pi_i \|_{TV}$ by means of eigenvalues and eigenvalue bounds.

Example 3.1. *Clockwise moving customers in homogeneous cycles* We consider the cycle of Theorem 1.1 with only customer 1 and the additional assumption of homogeneous nodes, i.e., $p_{\mathfrak{j}(1),\vartheta(1)} = p$ for all nodes. This means that the service times at any node of customer 1 are geometrically distributed with parameter p.
The behaviour of this cycle is stochastically equivalent to a *clockwise moving delayed random walk* on $\{1, \ldots, N\}$ described by a Markov Chain $X = (X_t : t \in \mathbb{N})$ with one-step transition matrix

$$P(j,m) = \begin{cases} p, & m = j \oplus 1 \\ 1-p, & m = j \\ 0 & \text{otherwise} \end{cases} \tag{3.1}$$

Its equilibrium distribution $(\pi(j) : j = 1, \ldots, N)$ is

$$\pi(j) = \frac{1}{N} \quad \forall j \in \{1, \ldots, N\} \tag{3.2}$$

We therefore shall, whenever possible without generating misinterpretation, delete the explicit reference to the customer's type in the state description.

3.1 Strong Stationary Times and Separation

Our first method is using strong stationary times to determine the rate of convergence for the network of Example 3.1, when the service probability is $p = 1/2$. For simplicity of the presentation we delete in this section the explicit reference to the customer's type in the state description.

Lemma 3.2. For the homogeneous cyclic network of Example 3.1 with nodes $1, \ldots, N$, N odd, and clockwise moving customer and one-step transition matrix

$$P(j,m) = \begin{cases} \frac{1}{2}, & m = j \oplus 1 \\ \frac{1}{2}, & m = j \\ 0 & \text{otherwise} \end{cases} \tag{3.3}$$

the total variation distance between $P^t(j,\bullet) = P(X_t = \bullet \mid X_0 = j)$ and the equilibrium distribution π can be bounded with $a = \frac{a\pi^2}{3}$ by

$$\| P^t(j,\bullet) - \pi \|_{TV} \leq 6\exp(\frac{-at}{N^2}).$$

The proof of the lemma utilizes separation distance as well as strong stationary times and their relation to total variation. The structure of the proof is as follows.

I. We bound total variation by separation.

Definition 3.3. *Separation distance*
The separation between two distributions Q_1 and Q_2 on state space E is

$$s(Q_1, Q_2) = \max_{\mathfrak{e}\in E}(1 - \frac{Q_1(\mathfrak{e})}{Q_2(\mathfrak{e})}). \tag{3.4}$$

For a homogeneous Markov chain X with initial state $\mathfrak{e}_0$ we denote the separation distance between the transient distribution $(P^t(\mathfrak{e}_0,\mathfrak{e}), \mathfrak{e} \in E) = (P(X_t = \mathfrak{e} \mid X_0 = \mathfrak{e}_0), \mathfrak{e} \in E)$ and equilibrium distribution π by

$$s(\mathfrak{e}_0, t) := s(P^t(\mathfrak{e}_0,\bullet), \pi) = \max_{\mathfrak{e}\in E}(1 - \frac{P^t(\mathfrak{e}_0,\mathfrak{e})}{\pi(\mathfrak{e})}). \tag{3.5}$$

Lemma 3.4. [Fil91]

$$0 \leq \|Q_1 - Q_2\|_{TV} \leq s(Q_1, Q_2) \leq 1 \tag{3.6}$$

II. We bound separation by strong stationary times.

Definition 3.5. A strong stationary time T is a randomized stopping time for a Markov chain $X = (X_n : n \geq 0)$ such that

1. $P(X_k = \mathfrak{e} \mid T \leq k) = \pi(\mathfrak{e}) \quad \forall k \in \mathbb{N}, \mathfrak{e} \in E$, or, equivalently,
2. $P(X_k = \mathfrak{e} \mid T = k) = \pi(\mathfrak{e}) \quad \forall k \in \mathbb{N}, \mathfrak{e} \in E$

That is, X_T has distribution π and is independent of T.

Proposition 3.6. 1. If T is a strong uniform time for $\{X_n\}$ then

$$s(\mathfrak{e}_0, t) \leq P(T > t), \quad t \geq 0. \tag{3.7}$$

2. Conversely, there exists a strong uniform time such that (3.7) holds with equality.

The tail distribution of the strong stationary time T provides an upper bound for the separation distance $s(\mathfrak{e}_0, t)$ between $P^t(\mathfrak{e}_0,\bullet)$ and π and thus for the total variation distance. It therefore suffices to construct a strong stationary time T for our cyclic IS network, and to bound its tail probabilities. We state the result in terms of the delayed clockwise moving random walker, who is just our customer of interest, see Example 3.1.

III. We bound strong stationary times explicitly.

Lemma 3.7. [AD86][Theorem 2 and Lemma 3] For the delayed random walk on the cycle $1, \ldots, N$, N odd, there exists a strong stationary times which can be constructed algorithmically such that for $N \geq 3$ and $t \geq N^2$

$$P(T > t) \leq 6\exp(\frac{-at}{N^2}) \qquad \text{with} \quad a = \frac{4\pi^2}{3} \tag{3.8}$$

Theorem 3.8 (Distance to equilibrium for 1-customer networks)**.** In the cycle of Theorem 1.1 with only customer 1 and homogeneous nodes, i.e., $p_{\mathrm{j}(1),\vartheta(1)} = p$ for all nodes an upper bound for the total variation distance between $P^t(j,\bullet) = P(X_t = \bullet \mid X_0 = j)$ and the equilibrium distribution π is

$$\| P^t(j,\bullet) - \pi \|_{TV} \leq 6\exp(\frac{-ak}{n^2}), \qquad \text{with} \quad a = \frac{4\pi^2}{3}.$$

The algorithmic construction of a stationary time

The algorithm for the construction of the strong stationary time for the cyclic 1-customer network is described in terms of the associated delayed random walk, see Example 3.1.
It follows the recipe developed by Aldous and Diaconis[4], who constructed a strong stationary time for a symmetric random walk on $1,\ldots,N$, N odd.
An analogous algorithm can be applied to the delayed random walk in Example 3.1, i.e. to a cycle of identical nodes, where we additionally assume that in each nodes service is finished within the current time slot with probability $\frac{1}{2}$.

The delayed clockwise moving random walk generates a sequence of pluses and zeroes – a plus for a step to the right, a zero for not leaving a position.
A strong stationary time for the random walk is now constructed as follows:

Algorithm 3.9.

1. For fixed integer $k \geq n-1$, define B_l as the set of binary k-tuples with l pluses (mod n). Let l^* be the index corresponding to set B_{l^*} with $\mid B_{l^*} \mid = \min_{l\geq 0} \mid B_l \mid$.
2. Partition the set of binary k-tuples into n groups of size $\mid B_{l^*} \mid$, the lth group being chosen arbitrarily from B_l. Define $\mathcal{T}$ as the resulting set of k-tuples.
3. Consider the sequence of successive steps in disjoint blocks of length k. Define the stopping time T as the first time that a block of k equals one of the patterns in $\mathcal{T}$.

The algorithm can be illustrated by applying it to $n = 5$ and $k = 4$:

Example 3.10.

1. For $l = 0,1,2,3,4$, the B_l are as follows:

$$\begin{aligned} B_0 &= (0000) \\ B_1 &= (+000),(0+00),(00+0),(000+) \\ B_2 &= (++00),(+0+0),(+00+),(0++0),(0+0+),(00++) \\ B_3 &= (+++0),(++0+),(+0++),(0+++) \\ B_4 &= (++++) \end{aligned} \tag{3.9}$$

   As both B_0 and B_4 consist of one 4-tuple, l^* can be chosen to be either 0 or 4.

2. As $\mid B_0 \mid = \mid B_4 \mid = 1$, choose one 4-tuple from each of the above B_l, $l = 0,1,2,3,4$. This might lead to

$$\mathcal{T} = \{(0000),(+000),(++00),(0+++),(++++)\} \tag{3.10}$$

Lemma 3.11. *T as constructed according to the previous algorithm is a strong stationary time for the cyclic 1-customer network of IS nodes.*

Proof. Analogous to that of Aldous and Diaconis[5] for the symmetric random walk. □

Seemingly, Aldous' and Diaconis' construction principle for strong stationary times for symmetric random walks does not apply to non-symmetric random walks, and to our delayed non-homogeneous random walks neither. In the following section, we therefore present different approaches to bounding the speed of convergence in networks.

[4] [AD86], Theorem 2
[5] [AD86]

4 Eigenvalue estimates for speed of convergence in cyclic IS networks

At present there seem to be no stationary times available for our network from Example 3.1. In the present section, we therefore shall discuss other methods which are suitable to bound the speed of convergence to equilibrium.
A classical method[6] is bounding the total variation distance between $P^t(\mathfrak{e}, \bullet)$ and π from above by means of formulas that depend on the second largest eigenvalue of the transition matrix or a related matrix. A classical example is

Proposition 4.1. Let $\pi(\mathfrak{e})$, $P(\mathfrak{e}, \mathfrak{e}')$ be a reversible Markov chain on a finite set E. Assume P is irreducible with eigenvalues $1 = \beta_0 > \beta_1 \geq \cdots \geq \beta_{N-1} \geq -1$. Then for all $\mathfrak{e} \in E$ and all $t \in \mathbb{N}$, we have

$$4 \parallel P^t(\mathfrak{e}, \bullet) - \pi \parallel^2_{\mathrm{Var}} \leq \frac{1 - \pi(\mathfrak{e})}{\pi(\mathfrak{e})} \beta_*^{2t}, \qquad \beta_* = \max(\beta_1, \mid \beta_{m-1} \mid). \tag{4.1}$$

4.1 Eigenvalue estimates for non-reversible transition matrices and 1-customer cyclic queues

For cycles of queues, reversibility usually does not hold. Then, if the transition matrix is non-reversible, related reversible matrices are relevant for estimates.
Let P be an ergodic transition matrix on a finite state space E with stationary distribution π. Then $\tilde{P}$, the time reversal of P, is defined as

$$\tilde{P}(x, y) := \frac{\pi(y) P(y, x)}{\pi(x)}. \tag{4.2}$$

The transition matrix P and its time reversal $\tilde{P}$ can the be combined into the multiplicative reversiblization $M(P)$ of P.

$$M(P) := P\tilde{P}. \tag{4.3}$$

It can be shown that $M(P)$ as defined in (4.3) is a reversible transition matrix [Fil91]. The eigenvalues of $M(P)$ are all real and nonnegative, and assumed to be arranged in decreasing order, i. e. $1 = \beta_0(M) \geq \beta_1(M) \geq \cdots \geq \beta_{N-1}(M)$, so that $\beta_1(M) \in [0, 1]$. $\beta_1(M)$, combined with the stationary distribution π, yields an upper bound for the total variation distance of the original chain.

Lemma 4.2. [Fil91] Let P be an ergodic transition matrix on a finite state space E and let π be the stationary distribution. Let $\beta_1(M)$ denote the second largest eigenvalue of the multiplicative reversiblization $M(P)$ of P. Then for any $\mathfrak{e} \in E$

$$4 \parallel P^t(\mathfrak{e}, \bullet) - \pi \parallel^2 \leq \frac{(\beta_1(M))^t}{\pi(\mathfrak{e})}$$

Often, the exact value of $\beta_1(M)$ is not known. We thus have to find upper bounds for the second largest eigenvalue of $M(P)$.

Theorem 4.3. For the homogeneous cyclic queue of IS nodes $1, \ldots, N$, the second largest eigenvalue $\beta_1(M)$ of the multiplicative reversiblization $M = P\tilde{(P)}$ can be bounded by

$$\beta_1(M) \leq 1 - \frac{24p(1-p)}{N^2 - 1}. \tag{4.4}$$

This yields (see [Fil91])

$$4 \parallel P^t(j, \bullet) - \pi \parallel^2 \leq N(1 - \frac{24p(1-p)}{N^2 - 1})^t.$$

[6] see, for instance, [Beh00], [Bre99], [DS91], [Fil91]

4.2 The Poincaré inequality

The proof of Lemma 4.3 relies on the computation of Poincaré and Cheeger constants.
To simplify the exposition, we suppose for this subsection that R is a given transition matrix reversible with respect to a stationary distribution π with $\pi(\mathfrak{e}) > 0 \quad \forall \mathfrak{e} \in E$. The following construction applies to $R = M(P)$ to give upper bounds on $\beta_1(M)$.(See, e.g., [DS91], Sect. 1).

The underlying graph of the Markov chain R is an undirected graph (possibly containing loops) with vertex set E; $\{\mathfrak{e}, \mathfrak{e}'\}$ is an edge if $Q(\mathfrak{e}, \mathfrak{e}') := \pi(\mathfrak{e}) \cdot R(\mathfrak{e}, \mathfrak{e}') > 0$.
Given two vertices $\mathfrak{e}, \mathfrak{e}'$ in the same connected component, we randomly choose a path (without repeated edges) as the canonical path $\Gamma(\mathfrak{e}, \mathfrak{e}')$ from $\mathfrak{e}$ to $\mathfrak{e}'$ (we assume that $\Gamma(\mathfrak{e}, \mathfrak{e}')$ does not contain loops) and define its length $\mid \Gamma(\mathfrak{e}, \mathfrak{e}') \mid$ as the number of its edges.
Set

$$K := \max_{x \text{ oriented edge}} \{(Q(x))^{-1} E[\sum_{(\mathfrak{e},\mathfrak{e}'):\Gamma(\mathfrak{e},\mathfrak{e}')\ni x} \mid \Gamma(\mathfrak{e}, \mathfrak{e}') \mid \pi(\mathfrak{e})\pi(\mathfrak{e}')]\} \tag{4.5}$$

if the graph is connected and $K = \infty$ otherwise.

Proposition 4.4. [DS91][Prop.1] *Poincaré inequality*
The second largest eigenvalue β_1 of a Markov chain R reversible with respect to a strictly positive stationary distribution π satisfies

$$\beta_1 \leq 1 - K^{-1} \tag{4.6}$$

with K the Poincaré constant defined by (4.5).

Corollary 4.5. In the Poincaré inequality, if we have in addition that R is irreducible, π is uniform, $R(\mathfrak{e}, \mathfrak{e}') = r$ whenever $\mathfrak{e} \neq \mathfrak{e}'$ and $R(\mathfrak{e}, \mathfrak{e}') > 0$ and no canonical path exceeds γ_* in length, then

$$\beta_1 \leq 1 - \frac{r \mid E \mid}{\gamma_* b} \tag{4.7}$$

with $b := \max_x$(expected number of canonical paths through x).

4.3 Cheeger's inequality

Another promising inequality for our further development is Cheeger's constant. Let P be a (not necessarily reversible) transition matrix with a strictly positive stationary distribution π.

Definition 4.6. *Cheeger constant*
Define the Cheeger constant as

$$h = h(P) := \min\{\frac{\sum_{\mathfrak{e}\in S} \sum_{\mathfrak{e}'\in S^c} \pi(\mathfrak{e})P(\mathfrak{e}, \mathfrak{e}')}{\pi(S)} : S \neq \emptyset \text{ and } \pi(S) \leq \frac{1}{2}\} \tag{4.8}$$

Proposition 4.7. The second largest eigenvalue β_1 of a transition matrix reversible with respect to a strictly positive stationary distribution π satisfies

$$1 - 2h \leq \beta_1 \leq 1 - \frac{h^2}{2} \tag{4.9}$$

If P is nonreversible, apply Cheeger's inequality, for instance, to $M(P)$.
For $M(P)$, we get with initial state $\mathfrak{e}_0 \in E$

$$4 \parallel (P^t(\mathfrak{e}_0, \mathfrak{e}) : \mathfrak{e} \in E) - \pi \parallel^2 \leq \chi_t^2 \leq (1 - \frac{1}{2}h^2(M))^t \chi_0^2 \tag{4.10}$$

with

$$\chi_t^2 := \sum_x \frac{((P^t(\mathfrak{e}_0, \mathfrak{e}) : \mathfrak{e} \in E) - \pi(\mathfrak{e}))^2}{\pi(\mathfrak{e})}. \tag{4.11}$$

5 Conclusions and further work

For a subclass of the celebrated BCMP networks we have derived several bounds on the speed of convergence to equilibrium. These bounds are particularly useful when it comes to deciding whether simulation runs have reached equilibrium, if we want to sample from simulated equilibrium quantities. This is of importance because even in the case of BCMP networks many important characteristics are not available explicitly.
A general procedure in our bounding algorithms is to prove that some theoretically obtained (more or less sharp) bounds which are not directly accessible can be bounded by quantities from investigations of related systems.
Our ongoing research is concentrated on other subclasses of BCMP networks.
In general, our investigation concentrates on discrete-time systems, which are of increasing importance for recent applications. But it is obvious that the results presented here have parallel statements in continuous time.

References

[AD86] Aldous, David and Diaconis, Persi: Shuffling Cards and Stopping Times, Amer. Math. Monthly 93, 1986, p. 333-348.

[AD87] Aldous, David and Diaconis, Persi: Strong Uniform Times and Finite Random Walks, Advances in Applied Mathematics 8, 1987, p. 69-97.

[BCMP75] Baskett, F., Chandy, K.M., Muntz, R.R. and Palacios, F.G.: Open, closed and mixed networks of queues with different classes of customers, in: JACM 22, 1975, nr. 2, pp. 248 - 260.

[Beh00] Behrends, E.: Introduction to Markov Chains. Braunschweig: Vieweg 2000.

[Bre99] Bremaud, P.: Markov Chains: Gibbs Fields, Monte Carlo Simulation, and Queues. Texts in Applied Mathematics (31). New York/Heidelberg/Berlin: Springer 1999.

[DS91] Diaconis, Persi and Stroock, Daniel: Geometric Bounds for Eigenvalues of Markov Chains, The Annals of Applied Probability, Vol. 1, No. 1, 1991, p. 36-61.

[Eck03] Eckert, Kersten: Bediensysteme in diskreter Zeit und ihre Netzwerke. Thesis, Department of Mathematics, Unversity of Hamburg, 2003.

[Fil91] Fill, James Allen: Eigenvalue Bounds on Convergence to Stationarity for Nonreversible Markov Chains, with an Application to the Exclusion Process, The Annals of Applied Probability, Vol. 1, No. 1, 1991, p. 62-87.

[Hen90] Henderson, W. und Taylor, P.G.: Product form in networks of queues with batch arrivals and batch services, Queueing Systems and Their Applications 6, 1990, S. 71-88.

[Hen91] Henderson, W. und Taylor, P.G.: Some new results on queueing networks with batch movement, Journal of Applied Probability, Vol. 28, 1991, S. 409-421.

[Klei75] Kleinrock, Leonard: Queueing Systems, Vol. I: Theory, New York/ Chichester/ Brisbane/ Toronto: Wiley-Interscience 1975.

[Miya94] Miyazawa, Masakiyo: On the characterization of departure rules for discrete-time queueing networks with batch movements and its applications, Queueing Systems and Their Applications Vol. 18, 1994, S. 149-166.

[TL07] Tippner, K. und Lorek, P: Speed of convergence for networks of queues, Preprint Department of Mathematics, University of Hamburg (in Preparation), 2007

Synchronised Concurrent Processes As A Modelling Pattern And Compositional Uniformisation: Work In Progress

Freimut Brenner, Bruno Müller-Clostermann
Institute for Computer Science and Business Information Systems (ICB)
Research Group *Systems Modelling*
University of Duisburg-Essen
45127 Essen, Germany
Freimut.Brenner@icb.uni-due.de, bmc@icb.uni-due.de
http://sysmod.icb.uni-due.de/

Abstract. This paper gives an overview of our current research centered around the recently published method of compositional uniformisation. This method allows to compute expected total times, e.g. the mean time to absorption, of a continuous time Markov chain which is the joint process of several independent absorbing Markov chains. Compositional uniformisation is briefly introduced and in addition to this reproductive part it is shown that the higher moments of the time to absorption are easily accessible by this method. We give a brief overview on performance and reliability modelling paradigms for concurrent processes and their mapping to a Markov chain which is built of marginal absorbing Markov chains. We discuss an extension to the field of reliability evaluation and sketch how compositional uniformisation can be applied to determine the time to failure of k-of-m systems. Finally, the application to Markov process algebra models is outlined.

Keywords. Concurrency, Markov process algebra, mean time to absorption, MTTF, uniformization

1 Introduction

To benefit from advances in research on new solution algorithms in addition to modelling tools the knowledge of typical modelling patterns is necessary to build meaningful and well-structured models. Like design patterns in software engineering a modelling pattern is a template that may be used repeatedly to build up more complex scenarios. A modelling pattern is a form of designing a model with a clearly stated intent, motivation and structure. For a certain modelling problem, the use of modelling patterns helps to develop models which are well-structured, readable, maintainable, transferable and amenable to further development. Obvious examples of modelling patterns include various types of concurrency, resource sharing, mutual exclusion, synchronisation, load distribution, scheduling, routing, branching, message passing, and more.

Here we consider the modelling pattern of synchronised concurrent processes, which is of outstanding importance in many application areas, of course including computer systems and communication networks. Examples from the field of timed Petri nets and extended queueing networks are used to illustrate this well-known concept; other modelling techniques like Markov Process Algebras and Performance Message Sequence Charts (PMSC) allow constructions following the same pattern.

It is obvious how to map a model consisting of m concurrent and independent (Markovian) processes, as described by these examples, to a Markov chain Y which is build from m marginal absorbing Markov chains Y_i. The synchronisation point of the model defines the global absorbing state of the Markov chain Y. Therefore the calculation of the mean response time (or cycle time if we assume a restart of all processes) is indeed identical to the calculation of the mean time to absorption.

Although quantitative evaluation techniques of models including this pattern are well established, there is still work to be done. A straight forward solution technique is to directly solve the global Markov chain Y; this approach is limited due to the state space explosion problem. In the given context the number of global states grows exponentially in the number of involved concurrent processes. Here we use the technique of compositional uniformisation which has been firstly described in [Bren07a]. The main intention of the current paper is to illustrate some application scopes of the new technique.

Related work has been done under consideration of the different modelling paradigms; here we name just a few [Bohn02], [HaKn02], [PEPA07].

The rest of the paper is organised as follows. In section 2 we introduce the modelling pattern of synchronised concurrent processes. In section 3 the technique of compositional uniformisation is briefly introduced. In sections 4 and 5 we sketch work in progress which is concerned with the application of this method to reliability and performance modelling.

2 Modelling Pattern: Synchronised Concurrent Processes

2.1 Basic Model

An important modelling pattern in many performance and reliability investigations are cyclic concurrent processes. Of particular interest are scenarios where a set of processes starts at the same instant of time, then proceed independently and finally synchronise in a shared event. Dependent on the application the synchronisation itself can be timeless or time-consuming. We want to derive the time duration until all processes or a subset of these processes have finished their progress. Note that in performance modelling the termination of a process means the completion of a task whereas in reliability modelling a process completion is associated with a component failure or the successful repair of a component.

The most popular techniques for the modelling of such processes are queueing networks, stochastic Petri nets, Markov chains, stochastic process algebras and discrete event simulation. Also system specifications given in SDL and MSC and more recently in UML have been used to derive performance models.

The basic pattern of cyclic concurrent processes which possess a common global synchronisation point is sketched in Fig. 1.

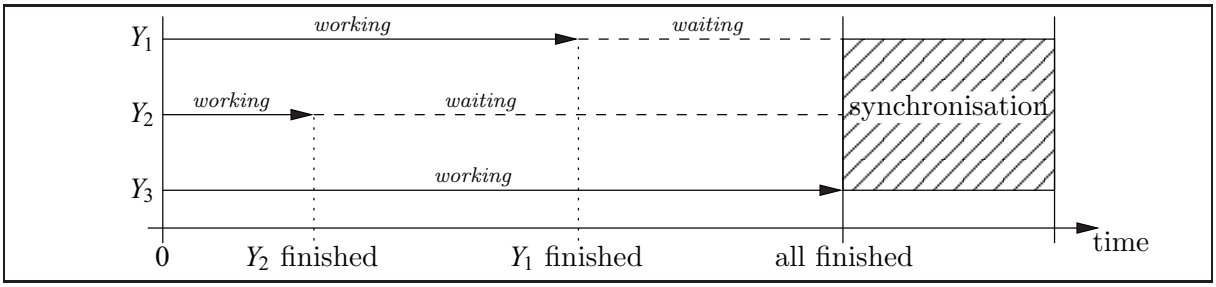


Figure 1: Three concurrent processes. Synchronisation begins when all processes have finished their work. Here, the synchronisation is time-consuming.

This basic pattern occurs in many performance and reliability models as illustrated in the next sections. The measure of interest here is the time until all processes do synchronise. Dependent on the application this measure may also be called response time, cycle time or time to absorption. In reliability evaluation we may be interested in the mean time until the $(m-k+1)$-th process out of m processes (or hardware devices) terminates which leads us to the mean time to failure (MTTF) of a k-of-m system.

We shortly review some modelling techniques by examples to demonstrate the most popular notions. All notions have in common that they may be mapped to a Markov chain Y with marginal absorbing Markov chains Y_i.

2.2 Timed Petri Nets, GSPN/DSPN

Fig. 2 shows a Petri net of concurrent processes. Each box represents a Petri net submodel of type GSPN or DSPN. A token at the output place of a subnet signals the termination (absorption) of the associated subprocess. If all output places are occupied another task may enter via the start-place.

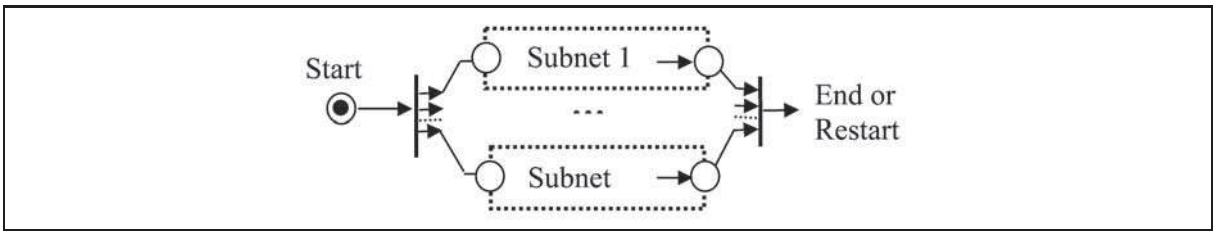


Figure 2: Timed Petri net model for synchronised concurrent processes.

2.3 Extended Queueing Networks, Fork/Join Networks

As an extension to product form queueing networks the class of extended queueing networks offers fork and join constructs. A fork construct creates a set of peer processes, which start immediately and proceed concurrently (and independently). The associated join construct collects all peer processes until the last one has finished. A recent paper [Arns06b] on the analysis of extended open fork/join queueing networks discusses different variations of this scheme including the two cases that the peer processes run through nodes each consisting of a single FCFS-Station (simple fork/join network) or through nodes which themselves may consist of fork/join networks (extended fork/join network). For approximate solutions of extended fork/join networks and related references, see [Arns06a, Arns06b].

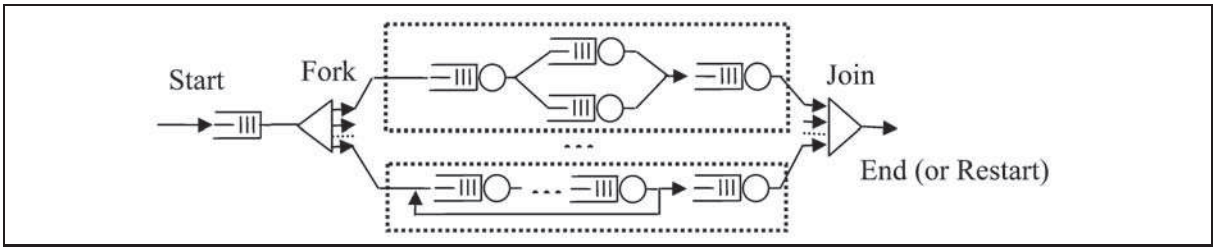


Figure 3: Extended Fork/Join Queueing Network (very general scheme).

In Fig. 3 we sketch a further extended scheme allowing arbitrary structures including branch and repeat, also nested fork/join structures can be allowed. An exact and efficient solution for this very general scheme is not feasible, however if we allow a new fork operation only if all running peer processes have terminated and the critical region is empty, the model is assessable by the new solution technique. We admit that the resulting model is not a true queueing network (there is no queueing!), nevertheless the interpretation as a queueing network and the existence of an efficient algorithm for a special case may inspire research towards solutions for more general cases.

2.4 Other Modelling Techniques

There are other modelling techniques which allow to describe synchronised concurrent processes. In the first place we have to mention stochastic process algebras where the synchronisation over shared events is the central concept. In case we use a common global event s as only synchronisation element, we obtain a model of m concurrent processes running independently until they synchronise over the event s.

Another technique are message sequence charts (MSC) which occur in different variations, e.g. in connection with SDL or as part of the UML. MSC is a high-level description of the message interaction between system components and their environment. Communication between distributed instances can be described by MSCs and these descriptions can be extended by notions for time consumption and resources and afterwards be included in a system performance model. There are several proposals how to enhance and analyse such performance enhanced MSCs using discrete event simulation or analytical queueing network algorithms, see e.g. [FLMS97], [HeMC03]. The scenario that a set of concurrent MSCs are initiated at the same moment models e.g. multicast or broadcast messaging in telecommunication systems; if we model the time consumptions either in the visited instances or over the communication lines by means of "Markovian" distributions (Exponential, Erlang, Hyperexponential), we yield the modelling pattern of concurrent processes. Again, the measure to be evaluated is the time until the last MSC has terminated.

3 Compositional Uniformisation

Here we explain the notion and the basic ideas of compositional uniformisation, for details we refer to [Bren07a, Bren07b]. Let $Y = (Y_1, \ldots, Y_m)$ be a CTMC, where the Y_i are independent absorbing CTMCs, and be H a random variable for the time to absorption of Y. The aim of this method is to compute the moments $\mathbb{E}[H^k]$ of the time to absorption and the expected total time $\mathbb{E}_A$ in some transient subset A of the state space of Y. The expected total time $\mathbb{E}_A$ is given by

$$\mathbb{E}_A = \int_0^\infty \mathbb{P}(Y(t) \in A) dt. \tag{1}$$

For the moments $\mathbb{E}[H^k]$, note that since H is a non-negative random variable, we have for $k \geq 1$: $\mathbb{P}(H > t) = \mathbb{P}(H^k > t^k)$. Hence, the following holds true

$$\begin{aligned} \mathbb{E}[H^k] &= \int_0^\infty \mathbb{P}(H^k > t) dt \\ &= \int_0^\infty k \cdot t^{k-1} \mathbb{P}(H^k > t^k) dt \\ &= \int_0^\infty k \cdot t^{k-1} \mathbb{P}(H > t) dt \\ &= \int_0^\infty k \cdot t^{k-1} (1 - \mathbb{P}(Y(t) \in S)) dt \end{aligned} \tag{2}$$

where S is the absorbing set of Y.

In the following algorithm the probabilities under the integral signs are reformulated such that the moments and $\mathbb{E}_A$ can be expressed by infinite sums, where each summand can be computed from the marginal CTMCs in a compositional way.

3.1 The Algorithm

Let $Y_i = (Y_i(t))_{t \in \mathbb{R}_{\geq 0}}$, $i = 1, \ldots, m$, be m independent continuous time homogeneous absorbing Markov chains. Let every Y_i be given by the finite state space E_i, the initial distribution $\nu_i(0)$, the set of absorbing states $S_i \subset E_i$,and the generator matrix $Q_i = (Q_i(j, \ell))_{j, \ell \in E_i}$.

The joint Markov chain $Y = (Y(t))_{t\in\mathbb{R}_{\geq 0}} := (Y_1,\ldots,Y_m)$ is given by the state space $E = \times_{i=1}^m E_i$, the initial distribution $\otimes_{i=1}^m \nu_i(0)$, the set of absorbing states $S = \times_{i=1}^m S_i$ and the generator $Q = \oplus_{i=1}^m Q_i$. That means the joint Markov chain has become absorbed if all of the marginal Markov chains have become absorbed.

Let $\mathcal{A}_i \subseteq E_i$, $i = 1\ldots m$, be arbitrary subsets of the marginal state spaces and $\mathcal{A} := \times_{i=1}^m \mathcal{A}_i$.

Definition 1. For two discrete functions $f,g : \mathbb{N}_0 \to [0,1] \subset \mathbb{R}$ and constant values $q_f, q_g \in \mathbb{R}_{>0}$ assigned to these functions, define the $\star$-operator by

$$(f\star g)(n) = \sum_{k=0}^{n} \binom{n}{k} \frac{q_f^k q_g^{n-k}}{(q_f+q_g)^n} f(k)g(n-k)$$
$$\text{and} \quad q_{f\star g} = q_f + q_g.$$

We also write $\star_{i=1}^m f_i := f_1 \star \cdots \star f_m$.

Note that the terms $f(k)g(n-k)$ under the sum are weighted with binomial probabilities. Hence, by estimating lower and upper tail probabilities of the binomial distribution it is possible to determine lower and upper truncation indices for the sum, such that the resulting error does not exceed a given $\varepsilon > 0$. For details see [Bren07b].

Theorem 2. *For $i = 1\ldots m$, let $P_i := I + 1/q_i Q_i$ and $\nu_i(n) = \nu_i(n-1)P_i$, for $n \geq 1$, where $q_i \geq \max_j\{|Q_i(j,j)|\}$.*

Then, with $\nu_i[\mathcal{A}_i](n) = \nu_i(n)(\mathcal{A}_i)$, $i = 1\ldots m$, and $q := q_1 + \cdots + q_m$, we have

(i) $\mathbb{P}(Y(t) \in \mathcal{A}) = \sum_{n=0}^{\infty} \frac{(qt)^n}{n!} e^{-qt} \left(\star_{i=1}^m \nu_i[\mathcal{A}_i]\right)(n)$,

(ii) $\left(\star_{i=1}^m \nu_i[\mathcal{A}_i]\right)(n)$ *is the probability that Y, if uniformised with rate q, is in $\mathcal{A}$ in the $n-$th step.*

Proof. See [Bren07b]. □

Substituting (i) of theorem 2 in the equations (1) and (2) yields

$$\mathbb{E}_A = \frac{1}{q} \sum_{n=0}^{\infty} \left(\star_{i=1}^m \nu_i[A_i]\right)(n), \qquad \mathbb{E}[H^k] = \sum_{n=0}^{\infty} \frac{k}{q^k} \frac{(n+k-1)!}{n!} \left(1 - \left(\star_{i=1}^m \nu_i[S_i]\right)(n)\right)$$

In order to compute these two sums, a truncation index N must be introduced. Let d_i be the dimension of the matrix Q_i. Then the time complexity is in

$$O\left(N \sum_{i=1}^{m} d_i^2 + mN^2\right).$$

The first term $(N\sum_{i=1}^m d_i^2)$ results from processing the Y_i in isolation and the second term (mN^2) results from computing the values $\left(\star_{i=1}^m \nu_i[A_i]\right)(n)$ and $\left(\star_{i=1}^m \nu_i[S_i]\right)(n)$, for $n = 0,\ldots,N$.

3.2 Probabilistic Interpretation

In order to understand the structure of the $\star$-operator, consider the example $Y = (Y_1, Y_2)$, where Y_1 has uniformisation rate q_1 and Y_2 has uniformisation rate q_2. The sequence of steps of (Y_1, Y_2) is the superposition of two individual Poisson streams and has rate $q_1 + q_2$. Then, the following two facts (partly illustrated in Fig. 4), are clear:

(a) The event that out of n steps of (Y_1, Y_2), Y_1 has contributed k steps and Y_2 has contributed $n-k$ steps happens with the binomial probability $\binom{n}{k}\left(\frac{q_1}{q_1+q_2}\right)^k \left(\frac{q_2}{q_1+q_2}\right)^{n-k}$.

(b) $\nu_1(\mathcal{A}_1)(k)\nu_2(\mathcal{A}_2)(n-k)$ is the probability that $Y \in \mathcal{A}_1 \times \mathcal{A}_2$, provided that Y_1 has performed k steps and Y_2 has performed the remaining $n-k$ steps.

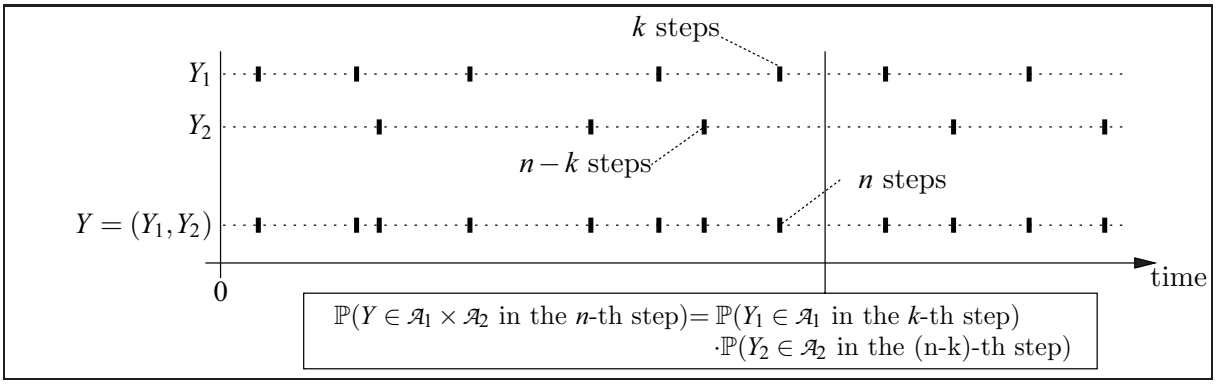


Figure 4: Marginal and joint steps and probabilities.

By the law of total probability the following sum is the (unconditioned) probability that $Y \in \mathcal{A}_1 \times \mathcal{A}_2$ in the n-th step.

$$(\nu_1 \star \nu_2)(n) = \sum_{n=0}^{\infty} \binom{n}{k} \left(\frac{q_1}{q_1+q_2}\right)^k \left(\frac{q_2}{q_1+q_2}\right)^{n-k} \nu_1(\mathcal{A}_1)(k)\nu_2(\mathcal{A}_2)(n-k).$$

It is clear that this line of argumentation can be generalised to more than two parallel CTMCs. On the one hand this explains the definition of the $\star$-operator and on the other hand we have just derived statement (ii) of theorem 2 (the experienced reader will immediately note that in theorem 2 statement (ii) implies statement (i)).

3.3 Numerical Example

From the previous sections it has become clear that all considered models can be mapped to a Markov chain which is assessable by compositional uniformisation. As a concrete example we solve a model with m concurrent processes as displayed in Fig. 5 in GSPN-notion. The submodels and hence the generator matrices Q_i are all identical.

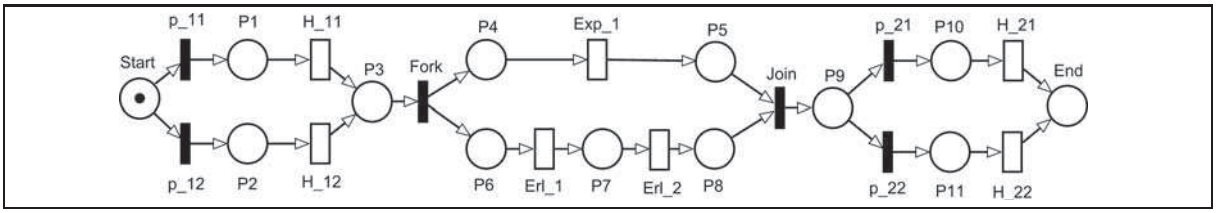


Figure 5: Absorbing Markov chain in GSPN-Notion

Immediately after start the process firstly runs through a hyper-exponential distribution (second order H_2-distribution), forks into concurrent exponential and a two-phase Erlang distribution, and finishes with a hyper-exponential distribution. As parameters for both H_2-distributions we choose $p_1 = 0.01$, $p_2 = 1 - p1 = 0.99$, $\mu_1 = 0.01$, $\mu_2 = 0.99$, yielding a mean value $\mathbb{E}[H_2] = 2.0$ and $CV[H_2] = \sqrt{49.5050} \approx 7.0359$. For the exp-distribution we use $\mu = 1.0$ and for the Erlang phases $\lambda = 2.0$. These distribution parameters define the entries in the resulting 10×10 generator matrices Q_i. Hence, the complete Markov model $Y = (Y_1, \ldots, Y_m)$ with generator matrix $Q = \oplus_{i=1}^{m} Q_i$ has 10^m states. We compute the mean cycle time (= mean response time = mean time to absorption = MTTA) for different values of m up to $m = 20$. Methods which directly operate on the matrix Q will work only for small values of m, since the overall model has 10^m states.

We emphasise that compositional uniformisation is not restricted to the case that all processes

	m=2	m=4	m=6	m=8	m=10	m=12	m=14	m=16	m=18	m=20
MTTA	8.293	12.975	17.164	21.102	24.871	28.507	32.032	35.459	38.798	42.054
comp. time [sec]	0.08	0.46	1.32	2.59	4.37	6.61	9.39	12.58	16.49	20.85

Table 1: Numerical Results for Example 1

are identical. Any kind of process can be included in the algorithm provided it can be represented as an absorbing Markov chain.

4 Work In Progress I: MTTF Of Generalised k-of-m Systems

A variation of the modelling pattern described in section 2 occurs with respect to k-of-m systems known from the field of reliability modelling and evaluation. When at least k units (= subsystems) out of m units are operational, the system remains operational; however, if $r = m-k+1$ units have failed, the complete system fails. As is the case with cyclic concurrent processes there is a time point which is a common starting point for all involved failure processes, each one representing the failure behaviour of a single unit.

Here we ask for the mean time to failure (MTTF), i.e. the expected time instant of the $(m-k+1)$-th failure. In case all components are described by an exponential failure rate λ there are well established results, e.g. for the 2-of-3 system the mean time to system failure is $5/6\lambda$. If the failure process is non-exponential the analysis gets more complex and a direct numerical solution will in general not be feasible. In particular if the failure processes are described by general phase distributions (including repairs) an approach which is based on the global state space will suffer from state space explosion.

More formally let H_i, $i = 1 \ldots m$, be iid. (identically and independently distributed) random variables for the time instants of failure of the m considered units. All H_i have the same distribution function $F(t)$. Define the random variable T as the time instant of the $(m-k+1)$-th failure, i.e. the event $\{T > t\}$ is determined by the equivalence

$$\{T > t\} \Longleftrightarrow \{\text{at most } m-k \text{ of the } H_i \text{ are } \leq t\}.$$

By the law of total probability one obtains

$$\begin{aligned}\mathbb{P}(T > t) &= \mathbb{P}(\textit{at most } m-k \textit{ of the } H_i s \textit{ are} \leq t) \\ &= \sum_{r=0}^{m-k} \mathbb{P}(\{\text{exactly } r \text{ of the } H_i \text{ are } \leq t\})\end{aligned}$$

For a fixed i, the event $\{H_i \leq t\}$ can be seen as the outcome of a Bernoulli trial with success probability $F(t)$. Hence, the probability that exactly r of the H_i are smaller than t is given by $\binom{m}{r}F(t)^r(1-F(t))^{m-r}$ (m Bernoulli trials). For a thorough treatment see [Triv82]. Consequently

$$\mathbb{P}(T > t) = \sum_{r=0}^{m-k} \binom{m}{r} F(t)^r (1-F(t))^{m-r} \quad \text{and} \quad \mathbb{E}[T] = \sum_{r=0}^{m-k} \binom{m}{r} \int_0^\infty F(t)^r (1-F(t))^{m-r} dt.$$

Now, assume the m units are modelled by absorbing Markov chains Y_i, with generator matrices Q_i and the set of absorbing states S_i. Let $S_i^{\complement} := E_i \setminus S_i$. Then for any i we have $F(t) = \mathbb{P}(Y_i(t) \in S_i)$ and $1-F(t) = \mathbb{P}(Y_i(t) \in S_i^{\complement})$. Furthermore, due to the iid property of the H_i we have $F(t)^r(1-F(t))^{m-r} = \mathbb{P}((Y_1,\ldots,Y_m)(t) \in (S_1,\ldots,S_r,S_{r+1}^{\complement},\ldots,S_m^{\complement}))$. Then

$$\mathbb{E}[T] = \sum_{r=0}^{m-k} \binom{m}{r} \int_0^\infty \mathbb{P}((Y_1,\ldots,Y_m)(t) \in (S_1,\ldots,S_r,S_{r+1}^{\complement},\ldots,S_m^{\complement}))dt.$$

Obviously, the integral is the expected total time, say $\mathbb{E}_{A(r)}$, which the Markov chain $(Y_1,\ldots,Y_m)$ spends in the set of states $A(r) := S_1 \times \ldots \times S_r \times S_{r+1}^{\complement} \times \ldots \times S_m^{\complement}$.

$$\mathbb{E}[T] = \sum_{r=0}^{m-k} \binom{m}{r} \mathbb{E}_{A(r)}. \tag{3}$$

The expected total time $\mathbb{E}_{A(r)}$ can be computed employing compositional uniformisation.

The method of compositional uniformisation is also applicable to compute the MTTF for heterogeneous systems, i.e. to k-of-m systems where the failure distribution of units $Y_1,\ldots,Y_m$ are not identical. In this case the terms under the sum in (3), including the binomial coefficient, will split in structurally different parts. This requires only some technical work.

Another field of application for the compositional uniformisation technique is the reliability evaluation of r resilient protocols. A protocol running on a system of m nodes is r resilient if it tolerates up to r node failures and still operates correctly. E.g. a byzantine agreement protocol is $\lfloor m/3 \rfloor$ resilient, because it can tolerate up to $\lfloor m/3 \rfloor$ failures. The MTTF-calculation for a r resilient protocol in case of non-exponential (and/or heterogeneous) node failure distributions is considered to be quite difficult [RaHT97]. We suppose that compositional uniformisation might provide new perspectives to quantitative evaluation of more complex cases.

5 Work In Progress II: Performance Measures Of Markov Process Algebra Models

Stochastic process algebras have become popular since the formalism was proposed by Herzog in [Herz90]. In particular, Markovian Process Algebras (MPAs) have drawn much attention due to the the fact that the quantitative solution of an MPA happens to be the solution of the underlying Markov chain. Examples for MPAs involve PEPA [Hill96], EMPA [BDG94, BeGo96], MTIPP [HeRe94] and IMC [Herm02]. Due to the nature of concurrency an MPA model might be subject to the state space explosion problem. Many approaches to combat this problem exist. For an overview the reader is referred to [Hill99].

Without going into the specifics of MPAs at this point we briefly sketch how the topic of absorbing joint Markov chains fits in the world of MPAs and how we plan to exploit it in future work.

Assume some MPA model which consists of three concurrent local processes Y_1, Y_2 and Y_3. The target quantity will be the steady state probability $\mathbb{P}(A)$ that the entire process is in the set A (assume that all conditions for the existence of such a steady state probability are met). The processes are allowed to do barrier synchronisations, i.e. every process is involved in every synchronisation. This is illustrated in Fig. 6. The processes start to evolve independently of each other. As soon as a local process is ready to synchronise, it must wait until all of the other processes become ready to synchronise on their part. After the synchronisation has taken place (in some state $\in \{s,s',s'',\ldots\}$) the processes again start to evolve independently of each other until the next synchronisation.

By now you will most probably have realised that a sample path of this system is a concatenation of several situations as described in Fig. 1. That means the following decomposition is possible: The entire system can be seen as a network of several subsystems $SUB(s)$, $SUB(s')$, $SUB(s'')$, ..., where the synchronising state x is the entrance state to the subsystem $SUB(x)$. In isolation a subsystem $SUB(x)$ can be modelled by a number m of absorbing Markov chains evolving in parallel. If $SUB(x)$ becomes absorbed in some state x', this means that in the entire system a transition from the subsystem $SUB(x)$ to the subsystem $SUB(x')$ would happen.

In order to calculate the steady state probability $\mathbb{P}(A)$ that the entire system is in the set A, we proceed as follows:

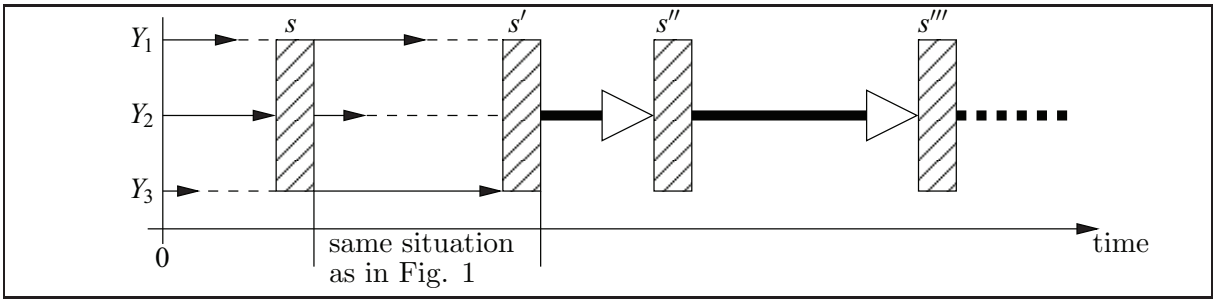


Figure 6: MPA model with barrier synchronisations. The thick arrows are an aggregated representation of the independent evolution of the marginal processes Y_1, Y_2 and Y_3. The global states in which synchronisations take place are denoted $s, s', s'', \ldots$.

(a) For every subsystem $SUB(x)$, calculate the expected total time $\mathbb{E}_A^{(x)}$ of being in A and the mean time to absorption $MTTA^{(x)}$. The mean time to absorption of subsystem $SUB(x)$ is the mean time that passes between entering and leaving this subsystem.

(b) Determine how the subsystems connect to one another. This boils down to solving an embedded discrete time Markov chain with state space $\{s, s', s'', \ldots\}$. Let $\pi = (\pi(s), \pi(s'), \ldots)$ be the steady state distribution of this DTMC.

Applying a well-known result from the theory of semi-regenerative processes (see [Çinl75]) the probability $\mathbb{P}(A)$ can then be calculated by

$$\mathbb{P}(A) = \frac{\sum_{x \in \{s,s',\ldots\}} \pi(x) \mathbb{E}_A^{(x)}}{\sum_{x \in \{s,s',\ldots\}} \pi(x) MTTA^{(x)}}.$$

In (a) the solution of the subsystems can be carried out by compositional uniformisation if A is the product of subsets of the marginal state spaces. In this case the state space explosion problem should not be an issue for a single subsystem. However, if the number of subsystems becomes large the approach sketched might be unfeasible. The same holds true for (b), since the number of subsystems coincides with the number of states of the embedded DTMC.

The basic idea of solving an MPA model in such a fashion goes back to [Bohn02]. Unfortunately, a method to compute the expected total times $\mathbb{E}_A^{(x)}$ in a compositional way was not available at that time.

Currently we are working at applying these ideas to the MPA PEPA. That means we are occupied with how to gain the subsystems from a given PEPA model and how to extract the embedded discrete time Markov chain.

6 Conclusion

We have shown how the new technique of compositional uniformisation can be applied to models which are built according to the modelling pattern of synchronised concurrent processes. Moreover we discussed some prospects of this technique in the field of reliability evaluation and performance modelling.

Acknowledgement

Special thanks go to Henrik Bohnenkamp who pointed out the relations in (2).

References

[Arns06a] Arns, M.: Approximative Verfahren auf erweiterten Fork/Join Warteschlangennetzen zur Analyse von Logistiknetzen (PhD thesis, University of Dortmund, 2006)

[Arns06b] Arns, M.: A New Aggregation Technique for the Analysis of Extended Open Fork/Join Queueing Networks by Decomposition (Proceedings of the 13th GI/ITG Conf. Measuring, Modelling and Evaluation of Comp. and Communication Systems, Nürnberg, 2006)

[BDG94] Bernardo, M.;Donatiello, L.; Gorrieri, R.: Modeling and Analyzing Concurrent Systems with MPA (Proceedings of the 2nd workshop on process algebras and performance modelling, vol. 27 Arbeitsberichte des IMMD, University of Erlangen-Nürnberg, 1994)

[BeGo96] Bernardo, M.; Gorrieri, R.: Extended Markovian Process Algebra (Proceedings of the 7th International Conference on Concurrency Theory, 1996)

[Bohn02] Bohnenkamp H.: Compositional Solution of Stochastic Process Algebra Models (PhD thesis, Department of Computer Science, Rheinisch-Westfälische Technische Hochschule Aachen, Germany, 2002)

[BGMT98] Bolch, G.; Greiner, S.; de Meer, H.; Trivedi, K. S.: Queueing Networks and Markov Chains - Modeling and Performance Evaluation with Computer Science Applications (John Wiley & sons, Inc.,1998)

[Bren07a] Brenner, F.: Cumulative measures of absorbing joint Markov chains and an application to Markovian process algebras (ICB-Research Report No. 12, University of Duisburg-Essen, 2007)

[Bren07b] Brenner, F.: Absorbing Joint Markov Chains: Exploiting Compositionality For Cumulative Measures (Proceedings of the 14th International Conference on Analytical and Sochastic Modelling Techniques and Applications (ASMTA 2007), Prague, Czech Republic, 2007)

[Buch06] Buchholz, P.: Structured Analysis Techniques for Large Markov Chains (SMC Tools, Pisa, Italy, 2006)

[Çinl75] Çinlar, E.: Introduction to Stochastic Processes (Prentice-Hall, Inc., Englewood Cliffs, N.J., 1975)

[FLMS97] Faltin, N.; Lambert, L.; Mitschele-Thiel, A.; Slomka, F.: PMSC - Performance Message Sequence Chart (Technical Report 10/97, IMMD VII, University of Erlangen-Nürnberg, 1997)

[HaKn02] Harrison, P. G.; Knottenbelt, W. J.: Passage Time Distributions in Large Markov Chains (ACM SIGMETRICS Performance Evaluation Review, Proc. of the 2002 ACM SIGMETRICS international conference on Measurement and modeling of computer systems SIGMETRICS '02, Volume 30 Issue 1, 2007)

[Herm02] Hermanns, H.: Interactive Markov Chains and the Quest for Quantified Quality (volume 2428 of Lecture Notes in Computer Science, Springer-Verlag, 2002)

[HeRe94] Hermanns, H.; Rettelbach, M.: Syntax, Semantics, Equivalences, and Axioms for MTIPP (Proceedings of the 2nd workshop on process algebras and performance modelling, vol. 27 Arbeitsberichte des IMMD, University of Erlangen-Nürnberg, 1994)

[Herz90] Herzog, U.: Formal Description, Time and Performance Analysis – A Framework (Technical Report 15/90, IMMD VII, University of Erlangen-Nürnberg, Germany, 1990)

[HeMC03] Hesham Kamal Arafat Mohamed; Müller-Clostermann, B.: Towards an Efficient Performance Evaluation of Communication Systems described by Message Sequence Charts (Forte 2003, Proceedings of the 23rd IFIP WG 6.1 International Conference Berlin Sept/Oct 2003, volume 2767 of Lecture Notes in Computer Science, pp. 413-429,2003)

[Hill96] Hillston, J.: A Compositional Approach to Performance Modelling (Distinguished Dissertations in Computer Science, vol. 12, Cambridge University Press, 1996)

[Hill99] Hillston, J.: Exploiting Structure in Solution: Decomposing Compositional Models (Performance Evaluation, 35, pages 171-192, 1999)

[PEPA07] Homepage of PEPA, http://www.dcs.ed.ac.uk/pepa/

[RaHT97] Rangrajan, S.; Huang, Y.; Tripathi, S. K.: On the scalability and mean-time to failure of k resilient protocols (Acta Informatica 54, 543-556, 1997)

[Triv82] Trivedi, K. S.: Probability & Statistics with Reliability, Queueing, and Computer Science Applications (Prentice-Hall, Inc., Englewood Cliffs, N.J., 1982)

A Software Architecture for Modular Implementation of Adaptive Protocol Stacks

Marc Schinnenburg, Ralf Pabst, Karsten Klagges, and Bernhard H. Walke
RWTH Aachen University, Faculty 6, Chair of Communication Networks (ComNets), Germany
{msg|pab|kks|walke}@comnets.rwth-aachen.de

***Abstract*— This paper presents a modular architecture for the design, implementation and performance evaluation of protocol stacks. The architecture allows a high degree of flexibility through composing complex behavior of a communication system using so-called Functional Units (FUs). The composition of Functional Units (FUs) to form a Functional Unit Network (FUN) is achieved by offering a generic management interface. We tackle the issue of protocol development for communication systems from the perspective of software design and -engineering. The requirements and the resulting interfaces for the Functional Units (FUs) are the main focus of this article. The proposed architecture is presented in the context of a reference model for multi-mode protocol stacks developed in the European Union-funded research project WINNER [1]. An exemplary implementation of the set of Data Link Layer functions as they are currently envisaged in WINNER and an implementation of an IEEE 802.16 (WiMAX) Data Link Layer (DLL) for performance evaluation is presented as a proof of concept.**

***Index Terms*— Adaptive Protocol Stack, Flexible Protocol, Multi-Mode, Functional Unit, Functional Unit Network, Modular Protocol**

I. Introduction

NEXT generation wireless communication system requirements target ubiquitous mobile access in a wide variety of application scenarios. These scenarios are differentiated by radio environment, usage, spectrum type or service requirements. In order to meet the full range of requirements such a system requires solutions tailored for the specific scenarios. Even though the individual solutions will be tailored for specific scenarios, a system concept would preferably embrace all these solutions being build on a common technology basis. This implies a flexible, adaptable air interface.

The WINNER project [2] has therefore developed a protocol reference model. The requirements of the above mentioned scenarios are met by different modes in this reference model. The reference model enables the efficient integration of multiple such modes in a complementary way, thereby allowing to take maximum benefit of the commonalities between the modes (see [3]). As a consequence, such a reference model requires protocols that conform to the given structure. In order to exploit commonalities between different modes of operation, the software of these protocols would (i) ideally follow a modular approach to allow a high degree of reusability and (ii) provide suitable structures and interfaces for the flexible composition of the individual modules.

[1] (**W**ireless World **In**itiative **Ne**w **R**adio), [1] is part of the 6th Framework Programme for Research

The increasing complexity in today's software systems has led to a number of new methods in software development in the last years. One goal of modern software design is to create systems consisting of preferably small units [4]. Every unit should have one cohesive responsibility, provided through a preferably slim and precise interface. The avoidance of tight coupling and the focus on testability leads to units that are easier to build, to test and to maintain. Thus the development and maintenance costs are reduced. Further, the quality of software is increased. As a second benefit, reusability of units is improved if dependencies between units can be kept to a minimum. The design paradigms from the software design domain can be immediately applied to protocol design for communication systems when a high degree of reusability of the units is targeted at, which is the basis for a flexible protocol architecture. A protocol stack consisting of small and independent units, here called Functional Units (FUs), with cohesive responsibility is therefore one of the key technologies for next generation radio networks.

To show the feasibility and gain a deeper understanding of the presented architecture, the research team of the Chair of Communication Networks (ComNets) has realized an implementation, which is used to model and evaluate new system proposals (e.g. as in WINNER) or evaluate proposals for the extension of existing systems (e.g. as in IST project FIREWORKS, where an Space Division Multiple Access (SDMA) extension has been integrated into the system concept of Worldwide Interoperability for Microwave Access (WiMAX)). Key performance indicators such as throughput per user or per station (e.g. Base Station (BS)) at different layers as well as packet delay at different layers can be evaluated.

The remainder of this paper is organised as follows: Section I-A of the introduction discusses previous and related work. Section II briefly introduces the context of the Modes Convergence Reference Model developed within WINNER. Section III describes the proposed FUs and the different aspects of their interfaces. Section IV outlines possible logical dependencies that may arise between different FUs and how to resolve these. Section V presents a short overview on how different data flows (e.g., in a BS) can be handled in our architecture. Finally, Section VI shows as an example of how the proposed architecture allows to implement the user-plane functions of the DLL

a) as they are currently envisaged by the WINNER project and how this implementation fits into the framework of the WINNER Modes Convergence Reference Model and

b) of an IEEE 802.16 (WiMAX) system.

The presented implementations serve as protocol stack prototypes for the simulative performance evaluation of the respective systems.

A. Related Work

The protocol architecture proposed in this work is inspired by both i) the work of Siebert and Walke in 2000/2001 (see [5] and [6]) and ii) the work of Berlemann et al. (see [7]). As proposed in these works, we adhere to the approach of complementing generic functionality with system-specific functionality to implement a certain protocol's (i.e., system's) behaviour. The approaches nevertheless differ in some important aspects: [5] and [6] propose rather coarse-grained modules [2] and concentrate on an efficient design of the protocols. Due to the coarse granularity, the solution is relatively monolithic and requires large efforts to enable run-time reconfiguration. [7] and our work focus on the composition of protocol layers out of smaller modules with a single cohesive functionality. The difference between the two is that Berlemann et al. investigate the effects of the composition (i.e., the behaviour of the composed protocols), while our work aims to define a software architecture and generic interfaces to enable the composition of arbitrary modules (FU). Additionally, this architecture offers the desired degree of flexibility to enable the run-time reconfiguration of protocols.

Following up on [5], Sachs in 2003 (see [8]) has taken up the idea of a generic protocol stack in focusing on a generic link layer for the cooperation of different access networks at the level of the DLL. However, not only DLL protocols but also adjacent layers' functions as for instance the control and management of the radio resources as well as mobility have to be considered in a multi-mode capable network. The work of [8] is continued by Koudouridis et al. in 2005 (see [9]) focusing on multi-radio transmission diversity and multi-radio multi-hop networking which can be seen as a use case for the software architecture elaborated in this paper.

Approaches that are related to the one presented in this work also exist in the domain of software engineering for operating systems. Much like [5] and [6], the *x-Kernel* architecture (see [10]) focuses on units that represent entire protocols and defines interfaces to connect them, while this work defines a framework to build up individual protocols from a 'toolbox' of atomic functions.

II. Modes Convergence Reference Model

The Modes Convergence Reference Model as proposed in [3] is designed to facilitate the coexistence and the cooperation of different modes in all logical nodes of the WINNER radio access network[3]. This efficient integration of multiple modes shall be termed 'Modes Convergence'. Modes Convergence-enabled devices will have the ability to dynamically adapt to different modes, giving them maximum flexibility and reducing complexity overhead compared to 'conventional' multi-mode devices (e.g. combined 3G / GSM handsets). Coexistence and Cooperation, i.e., Convergence of modes is desired in the following areas

Convergence in devices: This refers to the case of devices capable of (perhaps simultaneously) operating in different modes at the same time, such as multi-mode UTs, multi-mode BSs or multi-mode RNs. All these devices can benefit from a joint optimization of the functionalities in the different modes. The overall system design also benefits from a convergence of the modes, because the level of commonalities can be kept as high as possible if modes convergence is a design target from the early stages on.

Convergence in spectrum: At the current point in time, the exact spectrum allocation for beyond 3G (B3G) wireless broadband systems such as the WINNER system and the usage of this spectrum by the different envisaged modes or even systems is not known. Thus, the case of devices sharing the same available radio spectrum while operating in different WINNER modes can not be entirely ruled out. These devices could share the available spectrum efficiently if they apply a sharing algorithm that is common to the different modes or maybe common to different systems. Cooperation between modes may increase the overall efficiency of spectrum utilization and may contribute to interference mitigation.

Fig. 1 illustrates the high-level architecture of the multi-mode protocol stack for the flexible WINNER air-interface. The layer-by-layer separation into specific and generic parts (where appropriate) enables a protocol stack for multiple modes in an efficient way: The separation is the result of a design process where the identification and grouping of common (generic) functions is one of the main targets. The generic parts of a layer, marked '-g' in Fig. 1, can be identified on different levels, such as the Physical Layer (PHY), Medium Access Control (MAC) and Logical / Radio Link Control (LLC/RLC) as shown in the figure. The specific parts (marked '-r*n*') are reused in the different modes supported by the protocol stack. The composition of a layer out of generic and specific parts is exemplarily depicted in Fig. 1.

The management and the joint handling of the protocol stack operating in different modes are performed by the stack management, also shown in Fig. 1. When multiple modes are operated, this can be regarded as cross-stack management and it is envisaged to be performed by a stack modes convergence manager (Stack-MCM) which controls the management functionality in the respective protocol layers (N-Layer Modes Convergence Managers, (N)-MCM) in a hierarchical manner. For further reference on the proposed architecture, see [3]. The introduced multi-mode protocol reference model facilitates the structuring of an arbitrary layer into generic and specific parts. In providing guidance for understanding this structuring it marks up optimization potential in questioning the necessity of indicated differences. In this way, an increased protocol convergence is reached enabling an efficient multi-mode capable protocol stack for the WINNER system. This article takes the example of the DLL of the WINNER system to illustrate how such protocols can be composed from a set of mode- independent and mode-specific FUs.

[2]The modules are entire Layer 3 protocol entities, such as Call Control (CC), Mobility Management (MM) etc.

[3]Referred to as logical nodes are (i) the User Terminals (UTs) (ii) the BSs and (iii) the Relay Nodes (RNs)

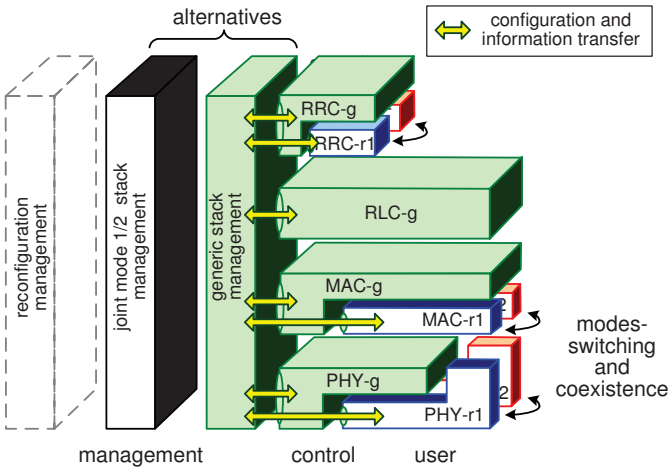


Fig. 1. The WINNER multi-mode protocol architecture, facilitating transition (switching) between modes (inter-mode handover) and coexistence of modes (e.g. in relay nodes connecting different modes)

III. Functional Units

As discussed in [3] the DLL of protocol stacks of wireless communication systems in general comprises among others the following set of functions: Automatic Repeat Request (ARQ), Segmentation And Reassembly (SAR), scheduling, multiplexing and buffering. If these units are supposed to be composed and connected in an arbitrary way, the necessity for a generalized interfaces arises. How should FUs be organised to support such a wide range of different tasks? How can these units be connected in a generic way to support the configuration of larger systems based on such units only? To answer these questions, we start analyzing the most fundamental requirements and describe interfaces that allow these requirements to be met. We will describe applications of the defined interfaces and how the FUs can be used to compose complex systems based on these interfaces. The set of functional units together with a description and configuration methodology is intended to form a toolbox for the implementation of protocols.

We have identified four interfaces which are at least necessary to realize an architecture as described in this paper. Namely, these interfaces are:

- Management
- Flow Control
- Data Handling
- Custom

These interfaces of a FU are depicted in Fig. 2. In the following sections these interfaces are described in more depth.

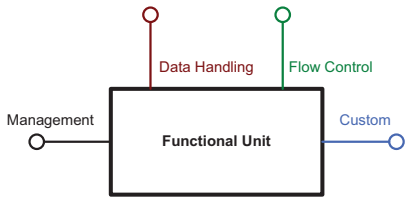


Fig. 2. General requirements of a FU

A. Interfaces

1) Data Handling Interface: The most fundamental requirement for FUs is the ability to handle data. In the following we will denote a basic data unit that is transmitted between FUs a *compound*. For now a compound can be seen as a chunk of data of variable size. FUs as part of a protocol stack may receive compounds for processing before and after such a compound has been transmitted over the air-interface. The first case is called outgoing data flow, while the latter case is referred to as incoming data flow. The interface for handling compounds has to provide services for accepting data in both directions, incoming and outgoing. The interface must further enable the FU to distinguish between compounds of both flows. To support that, it is advisable to choose two different methods: `sendData(Compound)` for compounds in the outgoing flow and `onData(Compound)` for compounds in the incoming flow as depicted in Fig. 3.

2) Flow Control Interface: In practice every FU has only a limited capacity to store compounds and often FUs do not need to store compounds at all to accomplish their task (e.g. forward error correction units). The physical layer on the other hand introduces a bottleneck, limiting the amount of information transmitted and thus the rate at which compounds must be handled. Without any flow control mechanism within a FUN, compounds could leave the FUN with much higher rates than the physical layer could possibly handle. This would result into a dropping of compounds in the physical layer. Buffering between the layers is not an adequate choice either, since the delay between processing the compound in the FUN and data transmission would increase. The increase of delay has several drawbacks.

First, timeout mechanisms would not work as expected. Retransmission timers could lead to retransmission of compounds although the last transmission of these compounds has not even been started. Such compounds would be added to the buffer several times, leading again to increasing delays.

Second, decisions of FUs based on feedback of the physical layer would loose accuracy; and gathered information would be outdated, when the consequences of the decisions would finally manifest.

Thus the need for an intra layer flow control arises. FUs must have the ability to prevent other units from delivering compounds to them, when they decide not to accept additional compounds. There are different reasons for a FU to decide not to accept compounds. All these reasons are direct consequences of the limited resources of the physical layer and thus only apply for *outgoing* flows. Resources in higher layers are usually not a bottleneck for *incoming* flows.

The implementation of this flow control mechanism is realized with two methods

- `isAccepting(Compound)`
- `wakeup()`

Before each `sendData` call the calling FU (whishing to send data) *must* make sure the target FU is accepting via the `isAccepting` method (see Fig. 3). It may also happen that a FU is asked if it is accepting further data and needs in turn to ask the next FU in charge, to return the right result. The

`wakeup` of a FU in turn is called by lower FUs to indicate that the lower (calling) FU is accepting data again.

A good example for this is a Stop-and-Wait ARQ protocol: the FU implementing the Stop-And-Wait ARQ is not accepting data if it is waiting for the acknowledgment of a transmission and is accepting data otherwise (assuming there is no buffer built into the ARQ). After the ARQ has received the acknowledgment for the transmission it can change its state from not accepting to accepting again and will in turn call the `wakeup` method of the upper FU to signal its state change. The upper FU can then try to send data to the ARQ again.

Note that flow control is only applied for outgoing flows. Flow control for incoming flows is not necessary. If a FU *A* cannot handle the data in the incoming flow due to limitations of the processing power of the device and if flow control would be available, a FU *B* located below *A* would need to store the data instead. Soon, the buffer of *B* would fill up and the FU below *B* would need to store additional data. Finally, if all FUs are not accepting any data in the incoming flow anymore the physical layer would need to store the incoming data and in the end start dropping incoming packets, too. Therefore, if a device cannot handle the data in the incoming flow, due to whatsoever limitations, respective flow control mechanisms between the sender and receiver need to be established to lower the data rate of the incoming data flow. Additionally, to support variations in data rate of the incoming data flow inherent to the system buffering to some extend can be applied in the FU of concern.

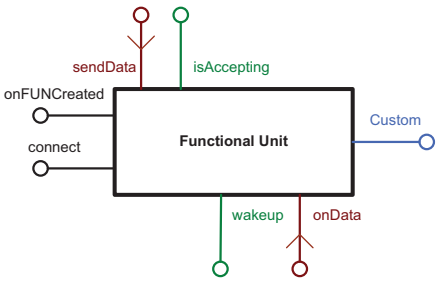


Fig. 3. Detailed interfaces of a FU

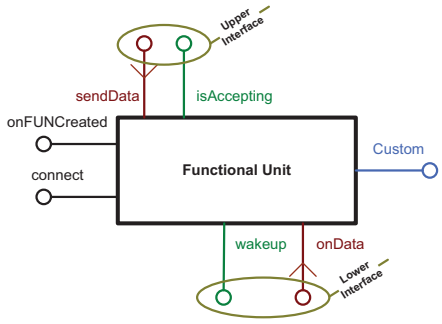
Remark: If no multiplexing and demultiplexing aspects are important, FUs are often drawn in a simplified version where the flow control and data handling interface are replaced by a logical upper and a lower interface (see Fig. 4).

3) Management Interface: The management interface of a FU offers the necessary functionality towards the FU Manager to manage the composition and (re-)configuration of the protocol stack or better a FUN[4]. Since this paper does not focus on the management of FUs, the interface presented in Fig. 3 does not show the complete set of functionality which is currently available. Two methods have been chosen as an example:

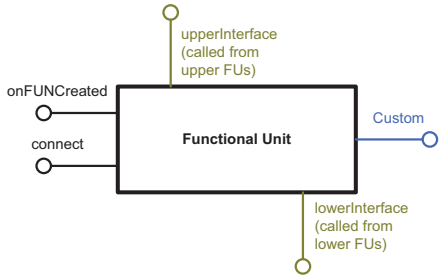
- `connect(FunctionalUnit)`
- `onFUNCreated()`

The `connect` method takes another FU as argument which should be connected to the FU in question. Special containers

[4]see III-B for more details on FUNs. Basically, a FUN is the framework which holds a number of interconnected FUs, forming a protocol with complex behavior



(a) `sendData` and `isAccpeting` can be logically combined into the upper interface of an FU. Accordingly, `onData` and `wakeup` form the lower interface



(b) FUs are often drawn only showing upper and lower interface to leave out unnecessary details

Fig. 4. FU with simplified (logical) interfaces

to support multiplexing and demultiplexing of data when the protocol stack is in full operation are maintained by this method.

`onFUNCreated` is a hook which is called by the surrounding framework of a FU to signal the successful creation of a FUN to the FU. Any special tasks the FU may need to undertake to get into a proper state for operation can be handled here. E.g. the FU can access other FUs of the FUN to resolve any dependencies it may have.

4) Custom Interface: All aforementioned interfaces are generic interfaces of a FU which need to be supported by each FU in order to ensure proper working as part of the described framework. However, it can be beneficial for FUs in terms of system performance (optimization aspects) to offer additional interfaces. These interfaces are summarized under the Custom interface.

The generic interfaces (described so far) don't export much information about the internal state of a FU (except whether a FU is accepting data or not through the `isAccepting` method). This contributes to one of the design targets formulated at the beginning: reduced coupling between FUs to increase reusability. For a further discussion on reusability and dependencies see section IV.

Fig. 5 and Fig. 6 show two examples for FUs that can have additional (custom) interfaces.

The method `retransmissionsPending` of the ARQ or `getLength` of the buffer can be used by other FUs of the FUN to optimize their operation. E.g. a FU can prefer a certain buffer over others if its length [5] exceeds a certain threshold.

[5]length is used synonym to size or fill level here

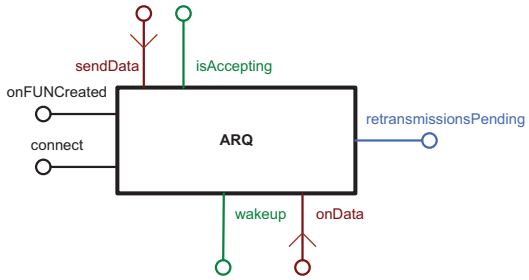


Fig. 5. FU with additional custom interface: ARQ

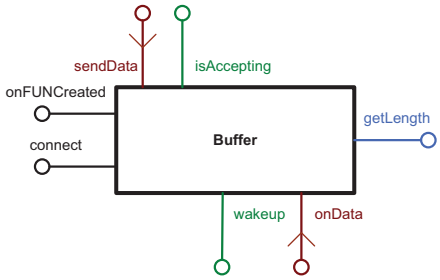


Fig. 6. FU with additional custom interface: Buffer

B. Functional Unit Networks

The methods `sendData` and `onData` are called by other FUs to propagate compounds through a Functional Unit Network (FUN). A simple setup of a FUN including the FU Manager is shown in Fig. 7. Three FUs are connected by the FU Manager to form a FUN. The outgoing data flow is from top to bottom (thus higher layers to lower layers). Every FU would forward the received data to its next lower FU without any choice. No (de)multiplexing aspects are present in this FUN. This is however only rarely the case within a protocol stack. Thus the frameworks needs means to handle (de)multiplexing.

Fig. 8 shows a part of a FUN with five FUs: two at the top, two at the bottom and one in the middle at a higher level of detail). The FU in the middle now needs to able to

a) multiplex outgoing data from upper FUs
b) demultiplex incoming data to upper FUs
c) demultiplex outgoing data to lower FUs.

Therefor, every FU contains three strategies for (de)multiplexing and according sets of references to other FUs: The Connector, the Deliverer and the Receptor (see Fig. 8). FUs call the `sendData` method of other FUs in their connector set to pass on compounds in the outgoing flow and call `onData` of FUs in their deliverer set to pass on compounds in the incoming flow. The strategy of Deliverer and Connector determine which FU will actually receive the compound. To wakeup upper FUs (see section III-A.2) the calls the Receptor which again forwards the call according to its strategy.

The FUs can be connected to multiple units in both directions to support multiplexing and scheduling, realized by choosing different strategies to select a unit for compound delivery. A FUN can now be constructed by choosing FUs from a toolbox of FUs and connecting them, defining their

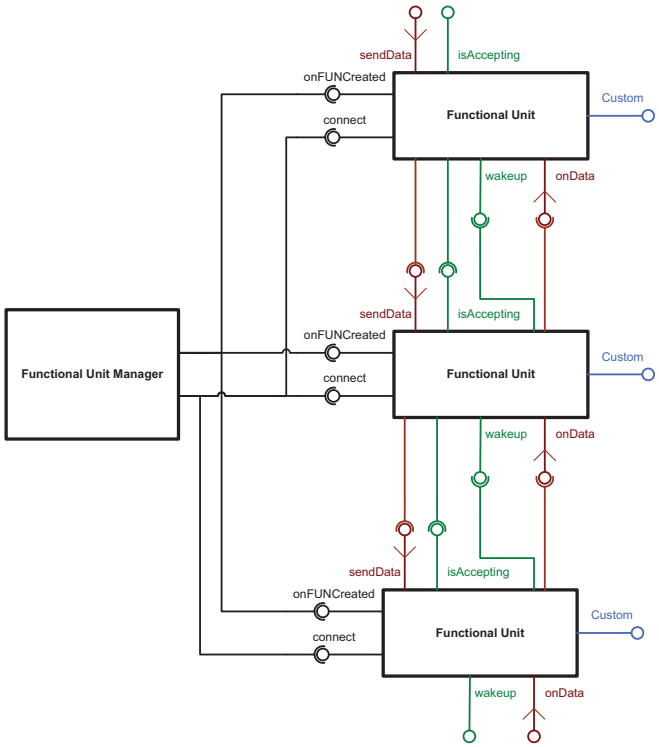


Fig. 7. Composition of FUs to form a simple FUN

connector and deliverer sets. It is possible to further identify a set of units as sink for outgoing flows: Compounds delivered to these units are leaving the FUN for delivery to lower layers. Another set of units may be identified as sink for incoming flows: Compounds delivered to these units are leaving the FUN for delivery to higher layers. Consequently, a FUN can be seen as a bi-directional data processing network. Input to the network is injected using either the `onData` or `sendData` method of any of the FUs. The output of the network is measurable at the sink units. The communication between peer entities of a protocol can be understood as data exchange between two FUNs.

C. Commands

Whenever a compound arrives in a FU, the FU gains control over the compound and can realize different behaviors by handling the compound accordingly. It may choose to mutate or drop the data unit, buffer it, forward it to other FUs or inject new data units into the FUN. A large class of FUs is characterized by enriching the compound, adding control information on outgoing compounds and reinterpreting the added information on incoming compounds. Usually these FUs provide a transparent connection to other FUs above. An ARQ protocol for example adds sequence numbers as control information to the compounds of the outgoing flow. It creates and injects compounds as acknowledgments in order to reply to compounds of the incoming flow. The ARQ instance in the peer FUN reinterprets the added control information, delivers valid information frames to some FU in the deliverer set and consumes dedicated compounds containing acknowledgments.

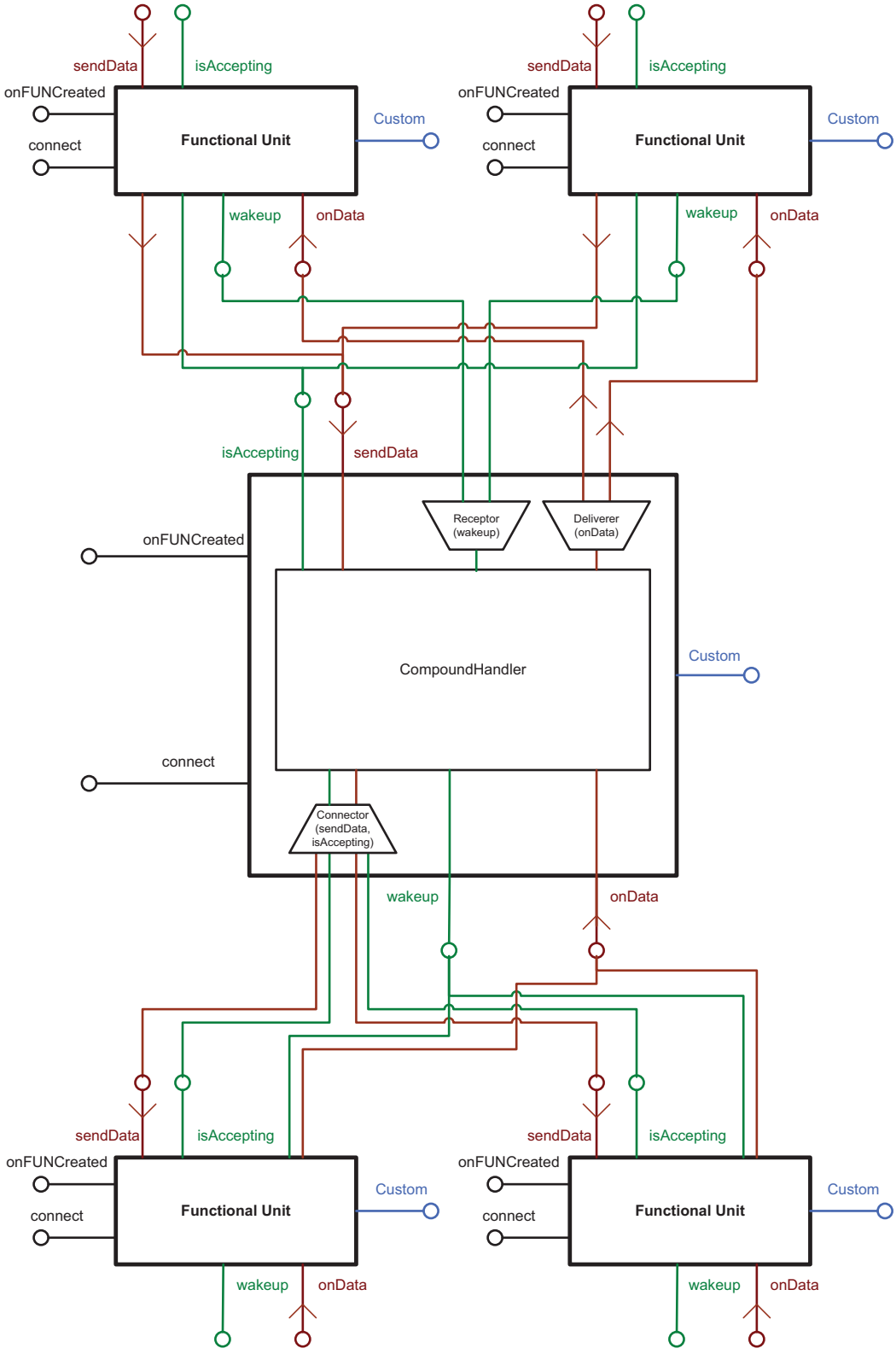


Fig. 8. FUN with FU at full level of detail

The control information added by FUs is called *command*. The command can have different characteristics for different purposes, like an information command or an acknowledgment command for the ARQ. The ARQ in our example is completely invisible to the FUs above. Even underlying FUs do not need to have knowledge about the control information added by ARQ implementations. The only FU that is required to be able to handle the ARQ command is the peer unit of

the ARQ. Sometimes, however, FUs add commands that are important to other FUs either in the peer FUN or within the same FUN. Connection identifiers may serve as an example for such information. FUs may require being able to retrieve the destination address of a compound which is part of a higher level (another FUs) routing command.

This leads to the requirement of having a possibility to access commands added by other FUs. Note that FUs cannot simply reinterpret control information added by other units to the compounds' data. FUs have no information about the layout of the FUN and therefore also have no information about the layout of the combined control information within the compound. There might be an arbitrary number of FUs in between the unit that added the control information and the unit that intends to access it. Additionally, the data might have been heavily modified by other FUs in between. The solution is to attach a set of commands to each compound. Since a FUN has a known number of connected FUs, there is a known set of potential commands.

The set containing all the commands of every FU within a FUN is called *command pool*. We can now specify more precisely that the *compound* as defined in Section III-A.1 is the union of a data unit and a command pool. Initially all commands within the command pool of a compound are inactive. The data attached to a compound is set to the data unit delivered by higher layers for transmission. A data unit is initially empty for compounds being created/injected in the FUN (e.g., ARQ acknowledgements). Parts of the command pool get activated during the propagation of a compound through the FUN, where every FU activates its command when in control. At the same time FUs can mutate the data.

A set of activated commands ordered by their time of activation is named a *command sequence*. A FUN is required to be free of cycles to assure that commands do not get activated more than once. Hence, an unambiguous command sequence must exist in which single commands may be retrieved.

1) Relaying of Compounds: The activation of commands from a non-extendable command pool introduces the problem of implementing relaying FUs [11]. Having a single point of activation implies that compounds may not cross the borders between the two FUNs from incoming to outgoing data flows. No FU may forward a compound using `sendData`, when it received the same compound via `onData`. This is a direct consequence from the activation of commands. Otherwise, it would be possible for the compound to be delivered to a FU that already activated its command. To implement relaying, the relaying FU has to inject a copy of the received compound, which has only those commands activated and copied that are in the command sequence before the relaying unit. The rest of the commands will remain inactivated. Fig. 9 gives an overview about the activation status of commands within a compound as it is passed from the originating FUN (left) via a FUN that has relaying capabilities (middle) to the final destination FUN (right). The relaying FU injects a copy of the incoming compound which holds a copy of the activated command of FU 'A' (and activates its own command 'B'), because the compound is not going to pass through 'A' in the outgouing path of the relaying FUN. Note that the relaying FU 'B' in Fig. 9 is transparent in the case of outgoing flows (left), for incoming compounds it can either perform the relaying as explained above (center) or decide not to further relay the compound because it has reached its final destination (right).

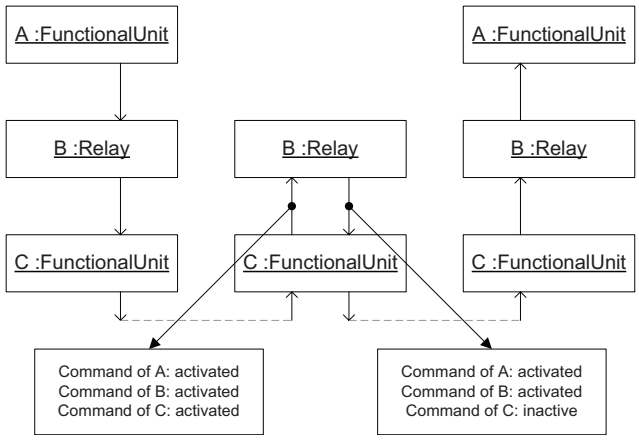


Fig. 9. Partial copy of a command pool for relaying

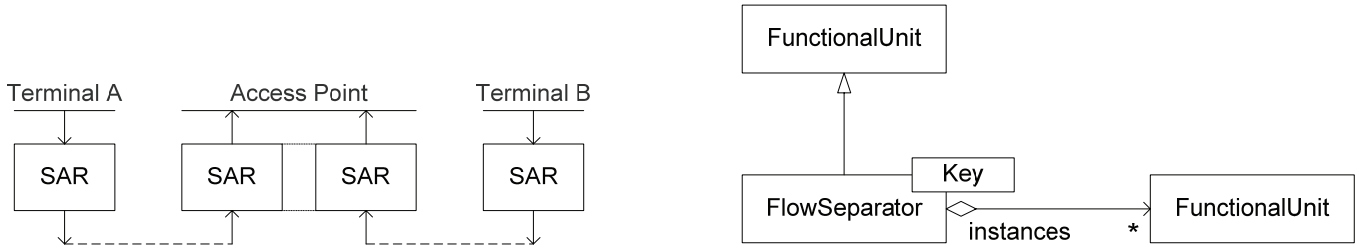
2) Coding of Commands: Besides commands being accessible by other FUs, delaying the coding of commands as part of the data has another advantage. Often information in communication protocols is not transmitted explicitly as a stream of bits, but implicitly through the choice of radio resources element like time, frequency, space or code. In a Time Division Multiple Access (TDMA) system for example with fixed slot reservations for connections, it would be useless to explicitly transmit connection identifiers. Nevertheless the information is indirectly transmitted through the choice of a specific slot. Such a slot must be chosen at some point of time based on the connection identity. A command provided by a connection aware FU may contain the connection identifier. But the choice how to transmit the connection identity is delayed, and the outcome may be different depending on the system. As we have shown, attributes of commands serve different purposes. Some are meant to be transmitted, while other attributes are only meaningful within a FUN and are not meant to be sent to the peer FUN. In order to be explicit about the purpose of a command attribute, we divide the attributes of commands in two distinct sets: The local on the one hand and the peer set on the other hand.

D. Five Aspects of a Functional Unit

To summarize the discussion above, we distinguish five aspects of a FU:

1) **Compound Handler:** implements the handling of compounds (Data Handling Interface) of a FU including intra FUN flow control (Flow Control Interface). The methods provided are
   - `onData(Compound)`,
   - `sendData(Compound)`,
   - `wakeup()` and
   - `isAccepting(Compound)` $\longrightarrow$ `Boolean`.

   Handling of compounds includes mutation, dropping, injection and forwarding. Activation and initialization of commands is considered as mutation.



(a) FUs of different flows

(b) UML Diagram for FlowSeparator in a FUN

Fig. 10. Flow Separation and dynamic instantiation of Functional Units

2) **Command Type Specifier:** defines the type of command provided by the FU. This type will be used to create an initial command pool and to verify unit dependencies as will be discussed in Section IV
3) **Connector:** holds the set of FUs that compounds will be delivered to in the outgoing direction; defines a strategy to select the appropriate FU for a given compound.
4) **Receptor:** holds the set of FUs in which the FU itself is in the connector set; defines a strategy to trigger outgoing data from the set of previous FUs (wakeup).
5) **Deliverer:** holds the set of FUs that compounds will be delivered to in the incoming direction; defines a strategy to select the appropriate FU to deliver a given compound.

IV. Unit Dependencies

Ideally a FUN would consist of FUs without any inter-unit dependencies. But that is not an option for building real world protocol stacks. Knowing what kinds of unit dependencies exist, what they imply and when to accept them is essential for the design of FUs and FUNs. We distinguish between two different kinds of unit dependencies: Direct and deferred coupling. **Direct coupling** is a dependency on the Custom *interface* of another FU (see section III-A.4); **deferred coupling** is the dependency on the *command* of another FU. When FU 'A' depends on the interface or the command of FU 'B' we say that 'B' is a friend of 'A'. Direct dependencies arise for example for

- multiplexing FUs that need assistance of their friends below them to decide where to deliver compounds.
- horizontal collaboration; FUs responsible for realising control plane functionality, receiving supervisory frames from a peer node and configuring their friends to modify the user data plane accordingly.
- vertical collaboration; layered protocol functions that must work close together but change behaviour in different places of a protocol stack.

To avoid tight coupling, those dependencies should rely on the most general interface possible [4]. The goal should be to make FUs depend on families of units sharing a common interface, and not to depend on a single type of FU. This allows friends to be exchanged without modifying the dependent FU. Since the exact layout of an FUN is unknown to the FUs, the FUN provides services for the FUs to find their friends by name and desired interface. Making the names of the friends a configuration option to dependent FUs results in a high degree of flexibility. Friends can be retrieved once after reconfiguration of the FUN. To retrieve a command from a command pool, the retrieving FU does not need to rely directly on an interface of the command's provider. It relies on the command's provider to be present in the FUN and on the type of command the provider specified.

There is another subtle difference between the direct coupling using the custom interface of a FU and the deferred coupling using the command attached to a compound. The interface, if used for information retrieval, yields a result which reflects the current state of the FU. The deferred coupling, using a command attached to the compound for, can only offer information reflecting the state of a FU at the time the compound passed through it.

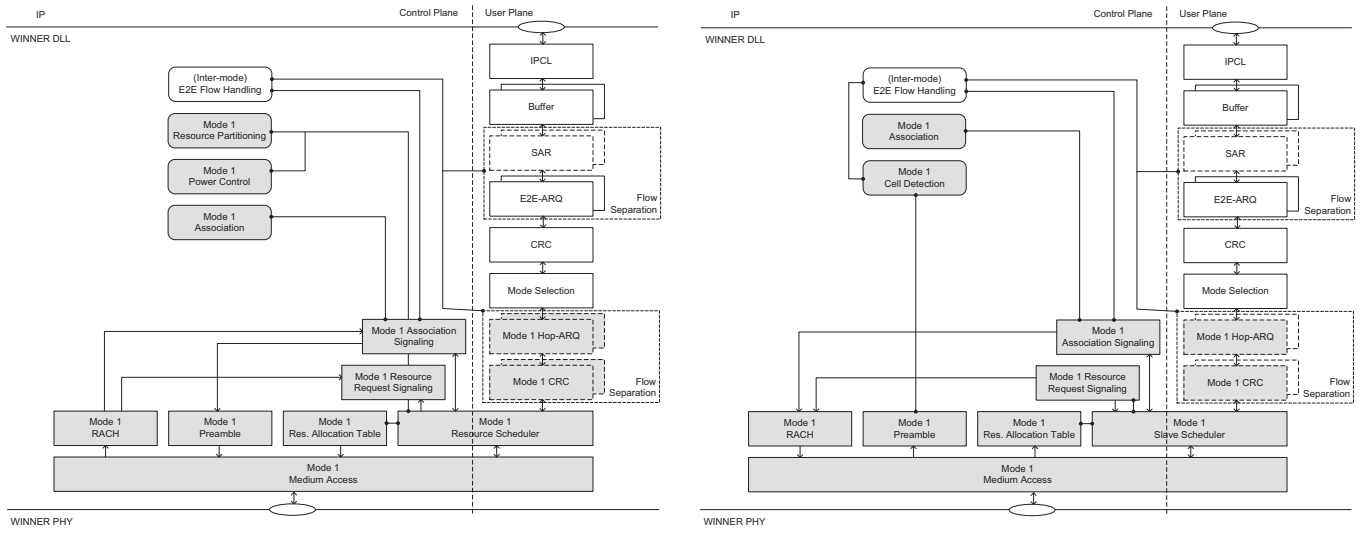
V. Flow Separation

FUs as described can have one or more internal states and exhibit a related behavior. A SAR unit for example needs to store segments of compounds to be able to apply segmentation and reassembly. A FUN therefore obviously requires different instances of a SAR unit for different peer FUNs as depicted in Fig. 10(a). To dynamically create FUs within a FUN depending on their data flow a so-called *flow separator* is proposed (see Fig. 10(b)). The flow separator itself is a FU, configured by a key to distinguish flows and a strategy to create FU instances. It is based on the *Instantiator* design pattern proposed by Gamma et al. in [12]. The compound handling including flow control is delegated to the according instance by the flow separator. This way, not each FU is burdened with task to be able to handle multiple flows, but a specialist was created that can be reused.

VI. Proof of Concept

A. Winner Protocol Stack

Fig. 11 exemplarily shows how the intended functionality of the DLL as it is currently discussed by the WINNER project can be composed out of a set of mode-independent FUs (white) and a number of mode-specific FUs (grey). An implementation of the DLL based on this concept is available as part of Wireless Network Simulator (WNS) for performance evaluation of the WINNER system. The DLL is depicted here only for one mode (Mode 1). However, in the WINNER project at least 2 modes are currently envisaged. It should



(a) in the Base Station

(b) in the User Terminal

Fig. 11. WINNER DLL as a Functional Unit Network

be noted that the mode-specific behaviour of the FUs shown in gray is achieved by pure parameterization, while identical implementations are being used when Mode 2 is used instead of Mode 1.

The used FUs can be further subdivided into three different classes:

1) The common, system-independent functions, these can be taken from a toolbox of generic protocol functions that can also be used to implement protocols for other, non-WINNER radio systems, examples are
   - ARQ units. The figure shows an E2E-ARQ for securing packets end-to-end over multiple radio hops and a Hop-ARQ that operates on a per-hop basis.
   - Buffers
   - Segmentation and Reassembly Units
2) The mode-independent, but WINNER-specific functions
   - IP Convergence Layer (IPCL)
   - Mode Selection
3) The WINNER-mode-specific functional units, which are shown as grey boxes. Example for such functions are
   - Resource Scheduler, which needs have detailed knowledge of the physical layer mode being used.
   - Hop-ARQ, which is likely to be a Hybrid ARQ and thus also closely linked to the physical layer mode being used.

The change of the parameterization of the mode-specific FUs to configure a new mode would be the responsibility of the entity that manages the respective protocol layer (see Fig. 1).

In addition to the discussed FUs Fig. 11 shows so-called Layer Services (drawn as boxes with round corners). These services, in contrast to the FUs, don't directly exchange information with some peer service in another protocol stack. They control, use and observe FUs within the protocol stack to fulfill a certain task. An example is the "Mode 1 Cell Detection" which simply observes the reception of Mode 1 Preambles by the respective FU. In a publish-subscribe manner the "Mode 1 Cell Detection" is subscribed to the "Mode 1 Preamble" reception. Again, this promotes minimizing dependencies between the modules of a layer to enhance reusablity (any other additional service or FU can subscribe to the "Mode 1 Preamble" reception) and flexibility (the strategy for the cell detection can easily be exchanged).

B. IEEE 802.16 Protocol Stack

The presented FU concept has also been used for an implementation of the IEEE 802.16 protocol stack. Fig. 12 shows the Functional Unit Network of an 802.16 protocol stack which realizes the BS and the Subscriber Station (SS) data plane and the OFDM PHY layer. As in the WINNER protocol stack, the 802.16 implementation uses several common FUs like the ARQ or the SAR. These generic FUs are drawn with white boxes. FUs that are modified or created for the FUN are drawn in grey boxes.

The function of the FUN can be divided into two classes, namely the Connection Control and the Radio Resource Control.

1) *Connection Control*
   The Classifier classifies compounds realizing flow separation for buffers, ARQ and SAR entities per CID. The Relay Mapper performs a rewrite of CIDs in the relay station for pairs of connections and a partial copy of the compound as shown in Fig. 9. As a result the relay implementation is not equipped with the FUs above the Relay Mapper except the buffers.
2) *Radio Resource Control*
   Radio Resource Management is mainly controlled by the Frame Builder which provides a common framework for frame based protocols. The frame is indirectly specified through the FUs that are directly attached to the upper side of the Frame Builder and through the

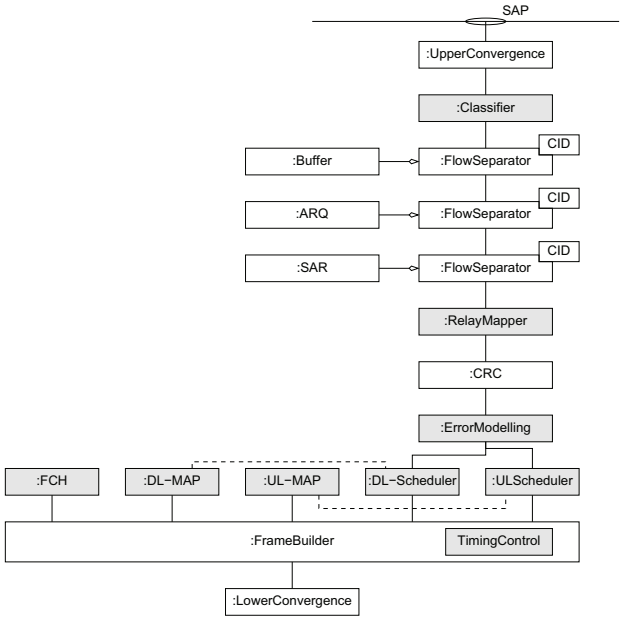


Fig. 12. Functional Unit Network of the IEEE 802.16 BS and SS

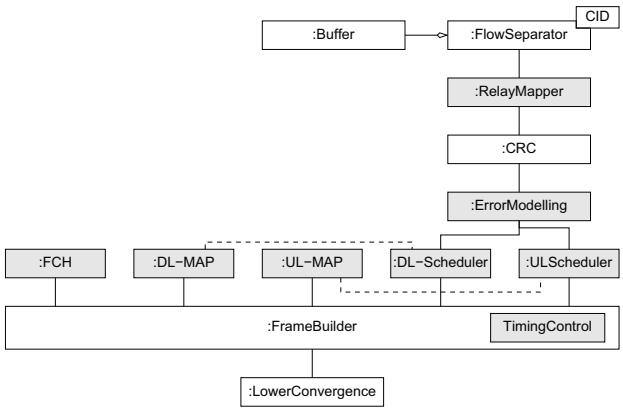


Fig. 13. Functional Unit Network of the IEEE 802.16 RS

Timing Control which is part of the Frame Builder. The Timing Control activates the attached FUs in a previously specified way. While the Frame Builder is a generic component, the Timing Control is system specific.

VII. Conclusion

We presented a software framework for flexibly building protocol stacks out of Functional Units (FUs). An implementation of this framework is used to evaluate the performance of wireless communication systems by means simulative performance evaluation. Simple, cohesive FUs are connected using a uniform interface to form arbitrary FUNs in order to build complex protocols. An interface, divided into the following five aspects, has been defined: Compound Handler, Command Type Specifier, Connector, Receptor and Deliverer. A solution for keeping dependencies between FUs at a minimum level is presented. The problem of flow separation is briefly discussed and a solution is sketched.

The presented framework opens up potential to

- Exploit similarities between the protocols of different wireless systems and hence facilitate efficient implementation of reconfigurable multi-radio devices
- Build systems based on multiple modes, each optimized to suit a specific environment, without the need to provide complete protocol software for each mode in the terminals and base stations.
- Implement software-defined radios where not only the physical layer but also higher layer protocols can be adapted through parameterization and configuration.
- Accelerate protocol stack development and performance evaluation.

Future work includes further investigations on the identification and implementation of reusable FUs conforming to the presented interface and on the management, parameterization and seamless reconfiguration of protocols represented through FUNs. Additionally, it is planned to release an implementation of the presented software architecture within the European research project *openWNS*, which currently applies for funding as part of the FP7. The *openWNS* project will release an open source system level simulator for the evaluation of wireless communication systems.

References

[1] http://www.ist-winner.org/.

[2] W. Mohr, "The winner (wireless world initiative new radio) project - development of a radio interface for systems beyond 3g." in *Proc. of IEEE Personal Indoor and Mobile Radio Conference 2005 (PIMRC05)*, Berlin, Germany, Sep 2005.

[3] L. Berlemann, R. Pabst, M. Schinnenburg, and B. Walke, "A flexible protocol stack for multi-mode convergence in a relay-based radio network architecture," in *Personal, Indoor and Mobile Radio Communications, 2005. PIMRC 2005. IEEE 16th International Symposium on*, vol. 2, 11-14 Sept. 2005, pp. 769–774Vol.2.

[4] H. Sutter and A. Alexandrescu, *C++ Coding Standards*. Addison-Wesley, 2005.

[5] M. Siebert, "Design of a generic protocol stack for an adaptive terminal," in *Proc. of 1st Karlsruhe Workshop on Software Radios*, Karlsruhe, Germany, Mar 2000, pp. 31–34.

[6] M. Siebert and B. Walke, "Design of generic and adaptive protocol software (dgaps)," in *Third Generation Wireless and Beyond (3Gwireless '01)*, vol. 0, San Francisco, US, Jun 2001. [Online]. Available: http://www.comnets.rwth-aachen.de

[7] L. Berlemann, R. Pabst, and B. Walke, "Multimode communication protocols enabling reconfigurable radios," *EURASIP Journal in Wireless Communications and Networking, Special Issue: Reconfigurable Radio for Future Generation Wireless Systems*, vol. 2005, no. 3, pp. 390–400, Aug 2005.

[8] J. Sachs, "A generic link layer for future generation wireless networking," in *Communications, 2003. ICC '03. IEEE International Conference on*, vol. 2, 11-15 May 2003, pp. 834–838vol.2.

[9] G. P. Koudouridis, R. Agüero, E. Alexandri, and et al., "Generic link layer functionality for multi-radio access networks," in *Proc. of 14th IST Mobile & Wireless Communications Summit*, Dresden, Germany, Jun 2005.

[10] N. Hutchinson and L. Peterson, "The x-kernel: An architecture for implementing network protocols," *Software Engineering, IEEE Transactions on*, vol. 17, no. 1, pp. 64–76, Jan. 1991.

[11] R. Pabst, B. Walke, D. C. Schultz, and et al., "Relay-based deployment concepts for wireless and mobile broadband radio," *IEEE Communications Magazine*, pp. 80–89, Sep 2004.

[12] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns : Elements of Reusable Object-Oriented Software*, professional ed. Boston: Addison-Wesley, 1994.

Automated Simulation of Communication Protocols Modeled in UML 2 with Syntony

Isabel Dietrich, Christoph Sommer, Falko Dressler, and Reinhard German

Computer Networks and Communication Systems
University of Erlangen-Nürnberg, Germany
{isabel.dietrich,christoph.sommer,dressler,german}@informatik.uni-erlangen.de

Abstract. This paper describes *Syntony*, an Eclipse-based framework that we developed for automated and tool-assisted development and analysis of network protocols. With the help of *Syntony*, we are able to use a simple graphical modeling language to describe complex protocols. In particular, we use UML 2 diagrams to construct simulation models to be executed in an event-driven simulation framework (currently, we are using OMNeT++). The translation of the UML 2 models is directly provided by *Syntony*. In addition to the use of standardized graphical modeling languages for developing and evaluating simulation models, the complete process of debugging and analyzing the protocol is tool-assisted. For verification purposes, we developed an UML 2 model of the communications within the AKTIV project Cooperative Cars (CoCar). We were able to demonstrate that *Syntony* can be used to generate executable simulation code and derive useful performance measures.

1 Introduction

Due to the complexity of today's networks, simulation is a widely used mechanism to evaluate the performance of new protocols or network configurations. However, the resulting simulation models are also very complex, often unclear or lacking documentation, and in most cases too complicated to be understandable at a glance. This leads not only to difficulties while debugging and maintaining the developed simulation models. The exchange of simulation models between research groups, and possibly between different simulators, is also hampered.

Fortunately, improved modeling techniques are available since several years. These techniques often include graphical mechanisms that enable humans to quickly understand the structure and behavior of the modeled system.

The Unified Modeling Language (UML) is such a graphical modeling language and is standardized by the Object Management Group (OMG). The current version of the standard is 2.1.1 [20]. A variety of academic and commercial tools exist that support model development with UML diagrams. To support the exchange and automated processing of UML diagrams, an alternate textual format called XML Metadata Interchange (XMI) has also been defined by the OMG.

Using the UML as a modeling language in the context of network simulation has several advantages compared to other available languages. The graphical representation of the main model elements and the enforcement of a modern, object-oriented approach to the modeling of systems make complex models far easier to comprehend. This also leads to the earlier discovery of conceptual errors in the model, accelerates model debugging and helps to locate errors sooner. In addition, the UML is very popular, easy to learn, and knowledge about it is already widely spread. Therefore, there is no need for another new programming language – and, in our case, there is no need to learn the internals of another new network simulator. And, of course, graphical programming is "en vogue" right now (and has been for the last years), which might increase the acceptance level among model developers.

For these reasons, we believe that the automated simulation of UML models is a very promising approach especially in the field of network simulation. This also takes UML one step further from a purely graphical formalism. The fact that the models remain machine-readable by means of the exchange format XMI leads to the applicability of UML for all kinds of model transformations. We expect that the development of new network protocols and the evaluation of their performance can be accelerated significantly with the adoption of the UML as the basis for network simulations. Therefore, we are developing an Eclipse-based tool that we named *Syntony*, which is capable of transforming UML models into executable simulation code.

Syntony is currently able to process standard-compliant UML models consisting of composite structure, state machine and activity diagrams. Performance annotations using the UML profile for QoS and Fault Tolerance Characteristics and Mechanisms, as well as some custom stereotypes are also possible.

To show the applicability and usage of *Syntony*, we developed a UML model of communications within the AKTIV project *CoCar* [2] and used *Syntony* to automatically generate an executable simulation model from it.

The remainder of this paper is organized as follows. After a brief overview of related work in Section 2, we show how computer networks can be modeled with the UML in Section 3. Section 4 discusses the inner workings of *Syntony* and details the transformation process from UML to simulation code. In Section 5, we present a case study using the CoCar scenario. Finally, Section 6 concludes the paper and gives some directions for future research.

2 Related work

Recently, a number of approaches have been published that use UML models to analyze the performance of software architectures. In most cases, UML models are not transformed into simulation, but into other mechanisms suitable for performance analysis, for example Petri nets, queuing networks, process algebras, or stochastic processes. Balsamo et al. [4] give a broad overview of some of the existing approaches. In their work, they mention two simulation-based

approaches, namely work by Arief and Speirs [3] and by De Miguel et al. [9]. Both approaches are included in the discussion below.

Of course, UML models have also been used as the basis for simulation studies. One of the earliest works in this area was done by Pooley et al. [14,21]. They use early versions of the UML 1 to generate discrete-event simulations from sequence diagrams. However, they maintain that state and activity diagrams are more suitable for system specification because sequence diagrams only capture specific message flows, but not all possible and legal message exchanges. They conclude that sequence diagrams are better suited for the animation of a simulation behavior than its specification. Arief and Speirs [3] transform systems specified with class and sequence diagrams into C++ or Java simulation code using a framework called *SimML*. Borshchev et al. [7] describe an approach that uses UML 1 state machines and composite structure diagrams, and annotations taken from the unofficial profile *UML for Real-Time*. From these models, they generate simulation programs in Java. They implemented their approach in the commercial tool AnyLogic. De Miguel et al. [9] define a set of custom stereotypes and tagged values to support the modeling of performance parameters. They specify systems using class, deployment and activity diagrams and annotate them with their custom profile. Their Simulation Model Generator (SMG) is able to transform these UML models into simulation models for the commercial simulator OPNET. Marzolla and Balsamo [5, 15, 16] use activity, use case, and deployment diagrams to specify systems and annotate performance aspects with the UML profile for schedulability, performance, and time (SPT). These models are transformed into code for a custom, process-oriented C++ simulator. Barth [6] uses activity and class diagrams to describe a system. Performance aspects are included from an external library and thus not modeled in UML. The system can be analyzed with a Java simulator. Michael et al. [17] developed a mapping of UML-RT models to the simulator OMNeT++. They specify the system using composite structure and state machine diagrams. Application requirements are specified with sequence diagrams which are generated from use cases. De Wet and Kritzinger [10] transform UML 2.0 models to Specification and Description Language (SDL) using the ITU Z.109 [13] profile. The ITU Z.109 profile defines a subset of the UML 2.0 and how the elements are mapped to SDL specifications. Existing methods for the incorporation of temporal aspects into SDL specifications are then used to analyze the model with process-oriented simulation. Choi et al. [8] use class and sequence diagrams to model systems. The sequence diagrams are then transformed into state machines. A discrete-event simulation model is generated from the classes and state machines. However, it is unclear how performance elements are integrated into this approach.

The main differences between *Syntony* and the related work discussed above are that our approach is fully compliant to the UML 2 standard, uses a set of diagrams that is particularly suitable for the modeling of network protocols (but also for most other applications), is built on widely known simulation tools and uses standardized UML profiles to annotate performance parameters.

3 Modeling systems and networks with UML 2

The UML offers a multitude of modeling elements and diagram types which can be used to model the structure and the behavior of systems. As the diagram types are partly redundant, a subset of the diagram types should be sufficient to model all relevant aspects of a system. The description of the system structure basically comprises the problem of which system elements there are, and how these elements are connected with each other. The possibilities to model system structure include a combination of component and deployment diagrams as in [11], or composite structure diagrams as in [7].

We decided to describe the system structure with *composite structure diagrams*. These diagrams can be used to display the internal structure of classes. This includes how other classes are nested inside a class, and how the nested classes can communicate via connectors attached to their ports.

The behavior of the entire system is determined by the functional operation of each system element, and the communication between the elements. There are three main options to model system behavior with UML 2 which have already been used in the literature. Activity diagrams are employed for example in [15] and [9], sequence diagrams in [3] and [8], and state machine diagrams in [7].

We chose to model the system behavior with *state machine diagrams*. UML state machines are very rich in features, which enables the modeler to produce very clearly structured, uncluttered models. At this point, there are two levels of detail to choose from. The less detailed variant is to annotate all transitions with transition probabilities. These probabilities can either come from measurements of an existing system, or from estimations. The detailed variant requires a complete specification of all transition effects and state actions (*entry*, *do*, *exit*). We use *activity diagrams* for their description. In this paper, we will only describe the second approach in detail.

The UML standard does not describe methods to model non-functional properties such as performance aspects of systems. However, UML profiles are defined as a flexible extension mechanism that can be used for this purpose. The OMG is currently in the process of standardizing a profile called Modeling and Analysis of Real-Time and Embedded Systems (MARTE) that will be suitable for the modeling of performance aspects. As the standardization is not yet finished, we had to rely on a combination of preliminary versions and own profile elements in this paper.

We defined a custom profile with three stereotypes. `<<simulationModule>>` can be used to reference existing simulation models in the UML model. Elements that are stereotyped as `<<simulationParameter>>` can be varied without recompiling the simulation, and can also be subject to systematical variation within given bounds. The stereotype `<<incrementStatistic>>` realizes a simple statistical counter.

The elements described above are sufficient to model arbitrary systems and networks. In particular with the multitude of UML 2 actions available for use in activity diagrams, it is assured that any behavior can be modeled using UML alone. However, this would become quite cumbersome as soon as the modeled

algorithms reach a certain complexity. It is therefore desirable to allow the usage of code in a textual programming language at least at certain places in a model. Appropriate UML elements for this are easily identified: OpaqueActions and OpaqueBehaviors allow the specification of a textual body and a corresponding language. We decided to support two different languages: the native language of the underlying simulation core (we are currently using C++), and the Object Action Language (OAL) [1] as a convenience language building on the UML action semantics standard [19]. OAL facilitates for example the specification of message sending and timer generation.

4 Syntony

We are developing the tool *Syntony* to enable simplified, flexible, and statistically sound simulation of models specified in the UML modeling language. As its input *Syntony* uses UML models as described in the previous section. The tool then analyzes the model, transforms it, and outputs a simulation model specified in C++ as is required by the used simulation core OMNeT++ [22]. The details of this process will be described in this section.

4.1 Basic concepts

Syntony is a software tool written in Java as a plug-in for the popular open source development environment Eclipse[1]. As such, its graphical user interface is realized as a set of Eclipse views. The input models have to be available in the XMI format as supported by the Eclipse UML 2 plug-in. Computer-Aided Software Engineering (CASE) tools able to export UML models into this format include for example the IBM Rational Software Modeler[2], Sparx Systems Enterprise Architect[3], and Omondo EclipseUML[4].

After the import of a particular UML model, *Syntony* transforms the model into C++ code. This transformation is described in detail below. The code is then compiled into a simulation program and executed. Apart from the initial import, these steps may be executed automatically. The execution of these steps is controlled from the *Tool Control* element of the graphical user interface (see Figure 1).

Another element of the user interface is the *Translation View* as depicted in Figure 2, which illustrates the structure of the input model and annotates error and warning messages generated during the transformation process.

4.2 OMNeT++

We currently rely on OMNeT++ [22] as the underlying simulation core. OMNeT++ is based on C++ and was designed to support efficient network simula-

[1] `http://www.eclipse.org/`
[2] `http://www.ibm.com/software/awdtools/modeler/swmodeler`
[3] `http://www.sparxsystems.com.au/`
[4] `http://www.omondo.com/`

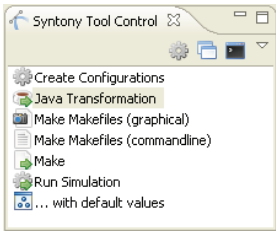


Fig. 1. Tool control view

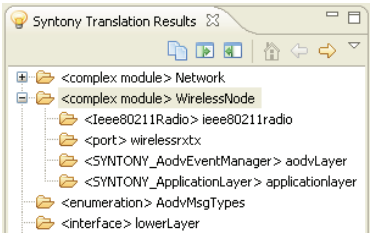


Fig. 2. Translation results view

tion. OMNeT++ distinguishes two kinds of classes, complex and simple modules. Classes that are composed of other classes, plus connections between them, are called complex modules. They are specified in the network description language (NED). Atomic classes with an associated behavior are called simple modules. They are written in C++, and are accompanied by a short NED description of their configuration parameters and interfaces, called *gates*, available for this class.

4.3 Transformation of UML model elements

Based on the above description of the OMNeT++ way of building simulations, it is quite clear how the main UML model elements correspond to OMNeT++ concepts. Classes with state machine diagrams become simple modules. Classes with composite structure diagrams are transformed into complex modules. UML ports are represented by a very similar construct in OMNeT++ called gates.

State machine diagrams are embedded in OMNeT++ simple modules. We do not use the Finite State Machine (FSM) mechanism built into OMNeT++ because UML features such as history states or orthogonal states would have been difficult to integrate with that mechanism. Instead, we base our translation of UML state machines on the *state* design pattern. A translation of some features of UML state machines into Java code using the *state* pattern has already been described in [18]. We loosely lean on that approach, with a few differences.

The main difference is that in the state pattern, the firing of transitions is delegated to the state objects. That is not the case in our implementation. Instead, we define a method for each state class that returns the currently enabled transitions depending on the current state, trigger and guards. The collection of enabled transitions is then evaluated at a central place, and the chosen transition is fired from there. The main advantage of this method is that the different firing priorities as defined in the UML standard, as well as extra tie-breaking rules, can be included a lot easier.

Figure 3 illustrates some of the features of UML state machines. Besides regular, initial, and final states, our implementation includes shallow and deep history states, nested states, orthogonal regions, joins and forks, junctions, and choice vertices. The figure also helps to demonstrate some pitfalls that can be

met when modeling state machines. Most of these pitfalls are mentioned as semantic variation points in the UML standard. The first issue is the question of what happens if an enclosing state does not have an initial state. This is the case with both regions contained in *State1* in the figure. Another question is what happens if several transitions are enabled at the same time. The standard defines that those transitions that leave the state with the deepest nesting level have the highest firing priority. In the figure, the transitions leaving *State11* and *State13* all have the same priority. If their triggers match and all guards evaluate to *true*, the standard does not define which one will be taken. *Syntony* currently chooses a random transition in this case, while a more sophisticated tie-breaking algorithm (or even a choice from several algorithms) would be desirable. Warnings or errors are created and shown in the *Translation View* if such a pitfall is recognized during the transformation.

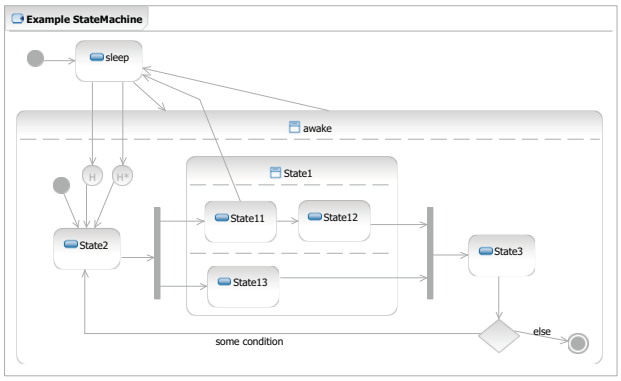


Fig. 3. Example of a state machine diagram

The effects of transitions and the entry, do, and exit actions of states may be specified in detail with activity diagrams.

All classes containing a composite structure diagram are translated to OMNeT++ complex modules. The parts contained in such a diagram are listed in the *submodules* section of the resulting NED file. The ports of the class are listed in the *gates* section, and the connections between class ports and ports on contained parts are detailed in the *connections* section.

Unfortunately, a few semantic issues concerning composite structure diagrams are left open in the UML standard. One such issue is the question of the exact connection topology for many-to-many connections. Possibilities to resolve this issue include the *star* pattern (connecting each element with all elements on the other side) and the *array* pattern (connecting the i-th element only with the i-th element on the other side). *Syntony* currently uses the star pattern, but outputs a warning message if this situation is encountered. Another problem refers to the semantics of signal forwarding. Imagine a signal arrives via some

connection at one side of a port, and should be forwarded to the other side of the port. What should happen if there are multiple connectors attached to that side of the port? The signal could be forwarded on all connections, just one connection, or a subset of the available connections. *Syntony* currently forwards the signal on all connections, and also outputs a warning message.

Syntony includes a compiler for OAL statements. This compiler has been written based on the EBNF production rules for OAL contained in [1] and the SableCC parser generator [12]. The compiler translates the statements from OAL to C++ and inserts them at the appropriate places in the C++ code generated by *Syntony*. Error messages are attached to the *Translation View* if a statement could not be translated.

4.4 Integrating existing simulation modules

The integration into an existing simulation framework inevitably raises the question if models that are already existing in the simulator can somehow be re-used in the context of our UML-driven simulations. The benefits of such a reuse are basically the same as for the reuse of other software components. Models that are already developed and tested need not be developed again. Models created by other people can be integrated. Performance comparisons are facilitated.

For the integration of existing OMNeT++ modules into UML models, we chose an approach that combines custom stereotypes with a model library representing the existing modules. In the model library, all reusable modules are represented by classes. The module parameters are attributes of these classes. All attributes are given appropriate default values. Elements from the model library can then be used in the context of a UML model by using them as parts in composite structure diagrams.

For the transformation into simulation code, two stereotypes have to be applied to the classes and their attributes. The `<<simulationModule>>` stereotype used for classed has one tag value which contains the OMNeT++ name of the module. During the transformation, parts that are stereotyped in this way are replaced by the corresponding OMNeT++ modules. Additionally, the module parameters have to be stereotyped as `<<simulationParameter>>`. This stereotype has several effects. For one, the corresponding attributes are placed in the OMNeT++ initialization file and can then be varied without recompiling the simulation. Second, the stereotype has tags that indicate how the stereotyped parameter should be varied systematically during the simulation runs.

5 Case study: Cooperative Cars

In order to further outline the capabilities of *Syntony* and to demonstrate the general feasibility of our approach, we created a model of the communications within the the AKTIV project *CoCar* in UML. Based on this model, selected details of the UML modeling process are outlined.

Project Cooperative Cars describes itself as follows [2]: The CoCar project is aiming at basic research for Car-to-Car (C2C) and Car-to-Infrastructure (C2I) communication for future cooperative vehicle applications using cellular mobile communication technologies. Five partners out of the telecommunications- and automotive industry develop platform independent communication protocols and innovative system components. They will be prototyped, implemented and validated in selected applications. Innovation perspectives and potential future network enhancements of cellular systems for supporting cooperative, intelligent vehicles will be identified and demonstrated. In a first step, potential application scenarios will be specified, data flows and information content analyzed, communication requirements of cellular C2C and C2I applications identified. Traffic and communication models will be worked out. A network load and latency simulator will be developed. The simulator behavior will be verified against a broad spectrum of telematics applications and related communication models. The simulation results will constitute the foundations for consecutive technical feasibility studies. The results shall also identify upcoming demands for cellular network evolutions. Extensible multi-party vehicular application protocols for global deployments will be worked out and evaluated in scope of the project.

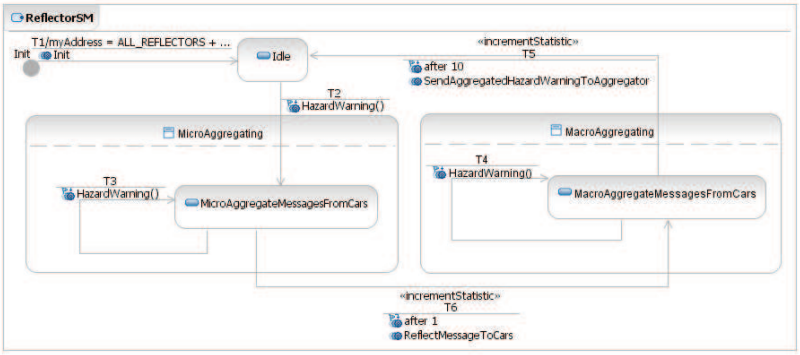


Fig. 4. State machine diagram for the Reflector component

The UML model we created represents a very early stage of the operations envisioned in the CoCar project. The behavior of all components is modeled with state machine diagrams. The state machine of one component, the *Reflector*, is shown in Figure 4. Also modeled in the simulated system are other network infrastructure components, as well as cars that communicate wirelessly via an air channel. The structure of the entire system is shown in Figure 5.

From this UML model, *Syntony* generates about 3000 lines of C++ code. While there are some redundancies in the code generated by *Syntony* and although this number also includes many debug statements (automatically generated from the names of states and actions), as well as statistical counters, it

is probably safe to say that a UML model (even a complex one) is easier to maintain and understand than several thousand lines of arbitrary code.

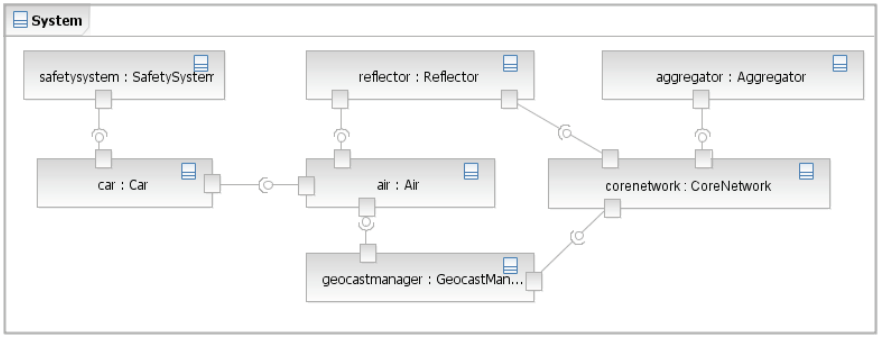


Fig. 5. The CoCar system

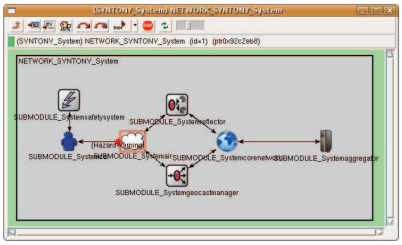


Fig. 6. Simulation model running in OMNeT++

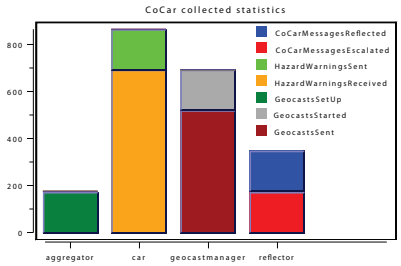


Fig. 7. Performance measures collected during the CoCar simulation

The translated code can then be directly compiled and executed as a model of the OMNeT++ simulation environment. A screenshot of the simulation execution is depicted in Figure 6. Using *Syntony* and OMNeT++, the UML model can thus be automatically simulated and typical measures such as the number of transmitted messages and other performance measures, depicted in Figure 7, can be collected.

6 Conclusion and future work

We demonstrated *Syntony*, an Eclipse-based tool for the automated translation of UML 2 models into executable simulation code. We also motivated the need

for standardized modeling languages for doing so and the choice of UML 2 diagrams that fulfill our requirements to a certain extent. Additional features can be modeled (and executed by *Syntony*) using profiles that allow to incorporate non-functional properties of the system, and by annotating modeled actions with OAL or C++ code. The usability of the tool chain has been verified by the development of a UML 2 model of the CoCar project's communication architecture.

This demonstrates that *Syntony* is well suited for large applications: it is flexible, easy to use, and can handle complex modeling tasks. As a consequence, existing UML tools can be used for the description of a complex simulation model. It is therefore possible to integrate simulation seamlessly in the system design process.

In conclusion, it can be said that *Syntony* supports very convenient graphical modeling and programming paradigms while achieving both similar accuracy and simulation performance compared to native models.

In future work, we plan to integrate the modeling facilities provided by the MARTE profile as soon as its standardization is finished. In particular, this will include the modeling of stochastic timing and probabilistic choices. Resolving the open semantic issues in this context will also be a part of our work.

We are also working on a standard-compliant action language as a replacement for the combination of OAL and C++ to increase the flexibility of the developed models. The integration of other simulation cores is planned as well.

In addition, we plan to extend the functional range of *Syntony* by adding facilities for the animation of the simulation execution, as well as a component for the integrated evaluation and presentation of simulation results. The inclusion of a component for statistical testing is also intended.

Acknowledgments. This work was supported by the Fraunhofer Institute for Integrated Circuits IIS, department for Communication Networks and by Vodafone Research & Development in the context of the CoCar project funded by the BMBF under grant number 01BU0692.

References

1. Accelerated Technology. Object Action Language Manual. Technical report, Embedded Systems Division of Mentor Graphics Corporation, 2004.
2. AKTIV Project. Fact Sheet: Adaptive and Cooperative Technologies for the Intelligent Traffic, November 2006.
3. L. B. Arief and N. A. Speirs. A UML Tool for an Automatic Generation of Simulation Programs. In *Workshop on Software and Performance (WOSP)*, pages 71–76, Ottawa, Ontario, Canada, 2000.
4. S. Balsamo, A. Di Marco, P. Inverardi, and M. Simeoni. Model-based performance prediction in software development: a survey. *IEEE Transactions on Software Engineering*, 30(5):295–310, May 2004.
5. S. Balsamo and M. Marzolla. Simulation Modeling of UML Software Architectures. In *European Simulation Multiconference*, Nottingham, UK, 2003.

6. M. Barth. Performance Assessment of Software Models In a Configurable Environment Simulator. In *International Conference on Software Engineering Research and Practice*, Las Vegas, Nevada, USA, 2003.
7. A. V. Borshchev, Y. B. Kolesov, and Y. B. Senichenkov. Java engine for UML based hybrid state machines. In *Winter Simulation Conference*, pages 1888–1894, Orlando, Florida, USA, 2000. Society for Computer Simulation International.
8. K. Choi, S. Jung, H. Kim, D.-H. Bae, and D. Lee. UML-based Modeling and Simulation Method for Mission-Critical Real-Time Embedded System Development. In *The IASTED Conference on Software Engineering*, Innsbruck, Austria, February 2006.
9. M. de Miguel, T. Lambolais, M. Hannouz, S. Betgé-Brezetz, and S. Piekarec. UML extensions for the specification and evaluation of latency constraints in architectural models. In *Workshop on Software and Performance (WOSP)*, pages 83–88, Ottawa, Ontario, Canada, 2000. ACM Press.
10. N. de Wet and P. Kritzinger. Using UML Models for the Performance Analysis of Network Systems. *Elsevier Computer Networks*, 49(5):627–642, 2005.
11. A. Di Marco and C. Mascolo. Performance analysis and prediction of physically mobile systems. In *6th international workshop on Software and performance*, pages 129–132, Buenes Aires, Argentina, 2007.
12. E. M. Gagnon and L. J. Hendren. SableCC, an Object-Oriented Compiler Framework. In *Technology of Object-Oriented Languages and Systems (TOOLS)*, page 140, 1998.
13. ITU-T. ITU Recommendation Z.109: SDL Combined with UML. Technical report, International Telecommunication Union, 2000.
14. C. Kabajunga and R. Pooley. Simulating UML Sequence Diagrams. In R. Pooley and N. Thomas, editors, *UK Performance Engineering Workshop*, pages 198–207, July 1998.
15. M. Marzolla. *Simulation-Based Performance Modeling of UML Software Architectures*. PhD thesis, Universit'a Ca Foscari di Venezia, 2004 2004.
16. M. Marzolla and S. Balsamo. UML-PSI: the UML Performance SImulator. In *1st International Conference on the Quantitative Evaluation of Systems (QEST)*, 2004.
17. J. B. Michael, M.-T. Shing, M. H. Miklaski, and J. D. Babbitt. Modeling and Simulation of System-of-Systems Timing Constraints with UML-RT and OMNeT++. In *IEEE International Workshop on Rapid System Prototyping (RSP'04)*, pages 202–209, Geneva, Switzerland, 2004. IEEE Computer Society.
18. I. A. Niaz and J. Tanaka. An Object-Oriented Approach To Generate Java Code From UML Statecharts. *International Journal of Computer and Information Science (IJCIS)*, 6(2):83–98, June 2005.
19. O. M. G. (OMG). Unified Modeling Language Specification (Action Semantics). Technical report, OMG, 2001.
20. O. M. G. (OMG). UML 2.1.1 Superstructure Specification. Technical report, OMG, 2007.
21. R. Pooley and P. King. The Unified Modelling Language and performance engineering. *IEE Proceedings Software*, 146(1):2–10, February 1999.
22. A. Varga. The OMNeT++ Discrete Event Simulation System. In *European Simulation Multiconference (ESM'2001)*, Prague, Czech Republic, 2001.

Traffic Anomaly Detection Using K-Means Clustering

Gerhard Münz, Sa Li, Georg Carle
Computer Networks and Internet
Wilhelm Schickard Institute for Computer Science
University of Tuebingen, Germany

***Abstract*—Data mining techniques make it possible to search large amounts of data for characteristic rules and patterns. If applied to network monitoring data recorded on a host or in a network, they can be used to detect intrusions, attacks and/or anomalies. This paper gives an introduction to Network Data Mining, i.e. the application of data mining methods to packet and flow data captured in a network, including a comparative overview of existing approaches. Furthermore, we present a novel flow-based anomaly detection scheme based on the K-mean clustering algorithm. Training data containing unlabeled flow records are separated into clusters of normal and anomalous traffic. The corresponding cluster centroids are used as patterns for computationally efficient distance-based detection of anomalies in new monitoring data. We provide a detailed description of the data mining and the anomaly detection processes, and present first experimental results.**

I. Introduction

Increasing processing and storage capacities of computer systems make it possible to record and store growing amounts of data in an inexpensive way. Even though more data potentially contains more information, it is often difficult to interpret a large amount of collected data and to extract new and interesting knowledge. The term data mining is used for methods and algorithms that allow analyzing data in order to find rules and patterns describing the characteristic properties of the data. Data mining techniques are attractive as they can be applied to any kind of data in order to learn more about hidden structures and correlations. However, this universality also has shortcoming because the generated knowledge does not have to be meaningful or useful. It is necessary to evaluate and interpret the data mining results with respect to a specific goal or purpose of the data analysis.

Intrusion detection systems (IDS) process large amounts of monitoring data. As an example, a host-based IDS examines log files on a computer (or host) in order to detect suspicious activities. A network-based IDS, on the other hand, searches network monitoring data for harmful packets or packet flows. In the late 1990s, progress in data mining research and the necessity to find better methods for network and host-based intrusion detection resulted in research activities attempting to deploy data mining techniques for anomaly and attack detection. Following this trend, the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining dedicated the KDD-CUP'99 competition to intrusion detection [1].

This paper focuses on *Network Data Mining* (NDM), i.e. the application of data mining methods to monitoring data recorded in computer networks. Section II gives an introduction to NDM and provides an overview and comparison of existing approaches. Most of them rely on rule learning algorithms.

In Section III, we present a novel NDM approach for anomaly detection based on the K-means clustering algorithm. The raw data consists of flow records that have been exported by routers and/or network monitors using Cisco Netflow [2] or the IPFIX protocol [3]. Flow records are easily available in many networks since flow monitoring techniques are already widely deployed for accounting purposes. We transform flow records into datasets with a small number of features for predefined time intervals and service-specific port numbers. Our goal is to identify time intervals showing anomalous traffic behavior, as it may be caused by network malfunctions or malicious attack traffic. The processing steps of our approach can be summarized as follows:

1) Training data containing flow records of both normal and anomalous traffic are transformed into feature datasets.
2) The datasets are divided into different clusters for normal and anomalous traffic using the K-means clustering algorithm.
3) The resulting cluster centroids are deployed for fast detection of anomalies in new monitoring data based on simple distance calculations.

While clustering monitoring data and identifying anomalies based on outlier detection has already been tried before, we are not aware of previous attempts generating additional clusters for anomalous traffic as we do.

Although a profound evaluation is still subject to ongoing work, we are already able to present some initial results in Section IV, demonstrating the general feasibility of the proposed anomaly detection method. Section V finally concludes the paper with an outlook on future work.

II. Network Data Mining

NDM may serve two different purposes. First, knowledge about the analyzed monitoring data is generated. This allows determining dominant characteristics and identifying outliers within the data records that can be considered as anomalous or suspicious. Secondly, NDM can be deployed to define rules or patterns that are typical for specific kinds of traffic,

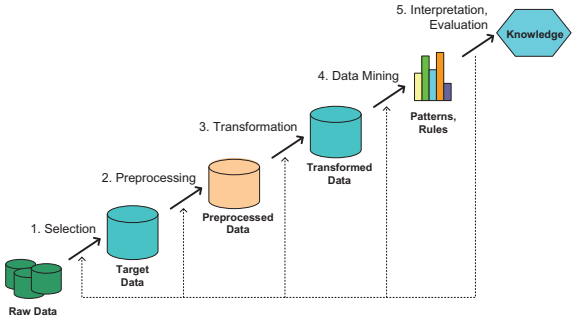


Fig. 1. Knowledge Discovery in Databases [4]

e.g. normal web traffic or traffic observed during a denial of service (DoS) attack. These rules and patterns can be used to analyze new sets of monitoring data and to check if these show similar properties and characteristics as the original data. Obvious applications profiting from such rules and patterns are network-based intrusion detection systems (NIDS) and traffic analyzers that characterize and classify traffic flows.

A. The KDD Model

NDM is applied to large amounts of monitoring data describing packet, flow, or connection attributes and statistics. These data are usually stored and processed in a database in order to retrieve the desired rules and patterns. Following the KDD (Knowledge Discovery in Databases) model [4] shown in Figure 1, five processing steps can be distinguished:

1) **Selection of raw data.** As an example, the data mining process can be restricted to monitoring data recorded in a specific period of time or observed at a certain monitor in the network.
2) **Data preprocessing.** Depending on the monitoring data and the data mining algorithm, it may be necessary or beneficial to perform cleaning and filtering of the data in order to avoid the generation of misleading or inappropriate rules or patterns. Disturbances can be noise, known changes such as dependencies on the time of day, missing data fields in some of the data records etc.
3) **Data transformation.** Applying data mining methods to raw monitoring data usually does not yield useful results since the numerous attributes within the original records are not equally relevant for the goal of the data mining task. Hence, a transformation has to be found that converts the raw data into datasets with a small number of relevant features. Also, it might be necessary to aggregate the data in order to decrease the number of datasets as well as the required memory and processing resources of the data mining algorithm.
4) **Data mining.** In this step, a data mining algorithm is applied in order to find rules or patterns.
5) **Interpretation and Evaluation.** It has to be evaluated if the data mining step generated useful results and which subset of the rules and patterns contains the most valuable information. In order to validate the rules and patterns, it might be necessary to repeat the process with a different selection of raw data.

As can be seen, the KDD model can be adopted to describe the necessary NDM processing steps at an abstract level although concrete realizations do not have to strictly follow this model. For example, some steps may be omitted and others performed in a slightly different order or jointly within in a single step.

B. Classification of Existing NDM Approaches

In this subsection, we present a selection of NDM approaches and compare them according to the following properties:

- **Type of raw data.** NDM can be applied to different types of monitoring data. Typical examples are packet data such as header information and/or payload, and statistical data about packet flows or connections including start and end time stamps, number of exchanged packets and bytes etc. Obviously, the data mining process as well as the achievable result depends very heavily on the type of raw data.
- **Extracted features.** The raw monitoring data is preprocessed and transformed in order to extract a set of relevant features. Extracting the set of most relevant features is essential for good data mining results.
- **Applied data mining algorithm.** A whole bunch of data mining algorithms have been presented in literature. They can be categorized according to the accomplished data mining task, e.g. clustering, classification, mining of association rules, characterization and discrimination etc.
- **Generated knowledge.** Finally, NDM approaches can be distinguished by the generated knowledge about the analyzed monitoring data and the capability to generate rules or patterns that can be applied to new sets of monitoring data.

In the following, we briefly describe the selected NDM approaches and compare them to our anomaly detection scheme. Table I summarizes the comparison based on the criteria listed above.

In the late 1990s, Lee and Stolfo studied different data mining methods for intrusion detection based on various kinds of monitoring and audit data. In the area of NDM, they used *tcpdump* [11] output as raw data and transformed it into connection records with the following features [5]: start time, duration, participating hosts, ports, the statistics of the connection, connection termination flag, and protocol (TCP or UDP). Additional attributes were added, e.g. the number of connections to the same host or service in the last w seconds. The rule-learning engine RIPPER [12] is applied to the training data with feature datasets labeled as normal or attack in order to generate rules for probing and denial-of-service attacks. These rules can later be used for misuse detection, i.e. in order to detect similar attacks in new monitoring data. However, they are not suitable for anomaly detection or the detection of new attacks showing a different behavior.

TABLE I
CLASSIFICATION OF NDM APPROACHES

Approach	Raw data	Features	DM algorithm	Knowledge
Lee & Stolfo (1998) [5]	Packets (tcpdump)	Connection records	Rule learning	Association rules and frequent episodes
Dokas et al. (2002) [6]	Packets (tcpdump) converted into TCP connection records	Time-based features	Outlier detection	Anomalies in analyzed data
Ertoz et al. (2003) [7]	Flow records (Cisco Netflow)	Time-based and connection-based features	Outlier detection	Anomalies in analyzed data
Esposito et al. (2005) [8]	Packets (tcpdump)	Connection records	Rule learning	Classification rules
Barbará et al. (2001) [9]	Packets (tcpdump)	Connection records	Rule learning	Association and classification rules
Luo et al. (2000) [10]	Packets (tcpdump)	Counters for TCP SYN/FIN/RST packets and number of different destination ports	Rule learning	Fuzzy association rules and fuzzy frequent episodes
Our approach	Flow records (Cisco Netflow, IPFIX)	Counters of bytes, packets, active flows for different time intervals and service-specific ports	K-means clustering	Centroids for normal and anomalous clusters

Researchers of the Computer Science Department at University of Minnesota performed similar studies applying rule-learning algorithms to connection records supplied as part of the KDD-CUP'99 competition [13] which consist of similar features as used by Lee and Stolfo. Later, they focused on outlier detection in order to detect anomalies based on time-based and connection-based feature sets [6], [14]. The LOF (local outlier factor) detection method [15] was selected to be integrated into the Minnesota Intrusion Detection System (MINDS) [7]. In MINDS, connection records ranked as highly anomalous by the LOF algorithm are further examined by an association pattern analysis module in order to generate rules and patterns for misuse detection. MINDS uses Cisco Netflow data as raw data of the data mining process.

Esposito et al. [8] deploy ToolDiag, a pattern recognition toolbox, to identify a small subset with the maximum discriminating power out of the connection features used by Lee and Stolfo. After, network behavior patterns are generated by applying a rule-learning algorithm to the selected features. Unfortunately, the authors do not disclose which connections features they finally used. Barbará et al. achieved very good results in the DARPA'99 intrusion detection test applying classification and association rules to connection records [9] . In [10], [16], the usage of fuzzy association rules and fuzzy frequent episodes is proposed.

In the next section, we present a novel NDM approach which is comparable to MINDS since it also uses flow records as input of the data mining process. However, we do not aim at classifying individual flows as normal or anomalous, but we use a smaller feature set in order to detect time intervals at which traffic anomalies occur. While MINDS groups normal flow records in a single cluster, we assume that normal and anomalous traffic form different clusters in the feature space. Therefore, we deploy K-means clustering to determine clusters for normal and anomalous traffic. Finally, we use a fast distance-based method to classify new monitoring data and detect outliers.

III. K-MEAN CLUSTERING OF MONITORING DATA

Our NDM approach deploys the K-mean clustering algorithm [17] in order to separate time intervals with normal and anomalous traffic in the training dataset. The resulting cluster centroids are then used for fast anomaly detection in new monitoring data.

In the next subsection, we describe the raw data and the extracted features that serve as input for the data mining algorithm. Thereafter, we explain the K-mean clustering algorithm and the resulting patterns. Finally, we show how the patterns can be used for classification and outlier detection.

A. Raw Data and Extracted Features

As input to the data mining process, we make use of flow records which are available in many networks due to the wide deployment of Cisco Netflow [2], [18] and compatible exporters. In this context, a flow is defined as a unidirectional stream of IP packets identified by a common IP five-tuple (protocol type, source IP address, destination IP address, source port, destination port). Apart from the IP five-tuple information, a flow record contains statistical information such as the number of packets and bytes observed in a certain period of time.

First, flow records are classified according to the transport protocol and predefined port numbers which are typical for commonly used services. As an example, TCP records with

source or destination port 80 are grouped as Web/HTTP traffic. The reason for this classification is that normal traffic looks very different depending on the service or application. Distinguishing flows by their protocol and service-specific port numbers thus allows applying the K-means clustering algorithm separately for different services identified by their (*protocol, port*) pairs. Flow records that do not fit into any of the predefined service classes are assigned to the corresponding default class for TCP, UDP, or ICMP traffic.

Separately for each class, we aggregate and transform flow records into datasets for equally spaced time intervals, considering the start time of each flow. The lengths of the time interval is chosen to be equal to or greater than the maximum expected flow duration in order to avoid that records of long-lasting flows cover multiple intervals. Note that the maximum flow duration is a configuration parameter of the router or network monitor exporting the flow records. Hence, setting this parameter to a small value (e.g. 1 minute or 10 seconds) allows generating datasets at a comparable small time scale. Each dataset contains the following features:

- *Total number of packets* sent from and to the given port in the considered time interval.
- *Total number of bytes* sent from and to the given port in the considered time interval.
- *Number of different source-destination pairs*[1] matching the given service-specific port and protocol and being observed in the considered time interval.

The motivation behind this feature selection is that the number of packets and bytes allows detecting anomalies in traffic volume, while the third feature helps detecting network and port scans as well as distributed attacks, which both result in an increased number of source-destination pairs.

The K-means clustering algorithm is applied to these datasets as explained in the next subsection.

B. K-means Clustering

K-means clustering [17] is a clustering analysis algorithm that groups objects based on their feature values into K disjoint clusters. Objects that are classified into the same cluster have similar feature values. K is a positive integer number specifying the number of clusters, and has to be given in advance. Here are the four steps of the K-means clustering algorithm:

1) Define the number of clusters K.
2) Initialize the K cluster centroids. This can be done by arbitrarily dividing all objects into K clusters, computing their centroids, and verifying that all centroids are different from each other. Alternatively, the centroids can be initialized to K arbitrarily chosen, different objects.
3) Iterate over all objects and compute the distances to the centroids of all clusters. Assign each object to the cluster with the nearest centroid.
4) Recalculate the centroids of both modified clusters.
5) Repeat step 3 until the centroids do not change any more.

[1] considering IP addresses and port numbers

A distance function is required in order to compute the distance (i.e. similarity) between two objects. The most commonly used distance function is the Euclidean one which is defined as:

$$d(x,y) = \sqrt{\sum_{i=1}^{m} (x_i - y_i)^2}$$

where $x = (x_1, ..., x_m)$ and $y = (y_1, ..., y_m)$ are two input vectors with m quantitative features. In the Euclidean distance function, all features contribute equally to the function value. However, since different features are usually measured with different metrics or at different scales, they must be normalized before applying the distance function.

An alternative to Euclidean distance is the Mahalanobis distance function that uses the inverse covariance matrix S^{-1} to reflect statistical correlations between different features:

$$d(x,y) = \sqrt{(x-y)^T S^{-1} (x-y)}$$

However, calculating and inverting the covariance matrix is computationally demanding for feature vectors with a large number of dimensions.

For initial evaluation of the proposed anomaly detection approach, we used a weighted Euclidean distance defined as:

$$d(x,y) = \sqrt{\sum_{i=1}^{m} \left(\frac{x_i - y_i}{s_i} \right)^2}$$

where s_i is an empirical normalization and weighing factor of the i-th feature. Note that the larger s_i, the smaller is the influence of the i-th feature on the distance. We found that good coefficients for the number of *packets*, *bytes*, and source-destination pairs (*src-dst*) are $s_{packets} = s_{bytes} = 5$ and $s_{src-dst} = 1$.

We apply the K-means clustering algorithm to training datasets which may contain normal and anomalous traffic without being labeled as such in advance. The rationale behind this approach is the assumption that normal and anomalous traffic form different clusters in the features space. Of course, the data may contain outliers which do not belong to a bigger cluster, yet this does not disturb the K-means clustering process as long as the number of outliers is small. As already mentioned, the clustering is done individually for the predefined services, identified by their typical *(protocol, port)* pair, as well as for the default classes that cover the remaining flows distinguished by the *protocol* value only.

The clustering algorithm divides the training data into K clusters, but does not determine if a cluster reflect time intervals of normal or anomalous traffic. This decision has to be made manually or by heuristics. For example, a higher average in the number of packets can be taken as an indicator for an anomalous cluster. It may occur that clusters are very close to each other. This can have several reasons: Either the number of clusters K has been badly chosen or the training data is very homogeneous, e.g. because it does not contain any anomalous traffic or because the anomalous traffic looks very

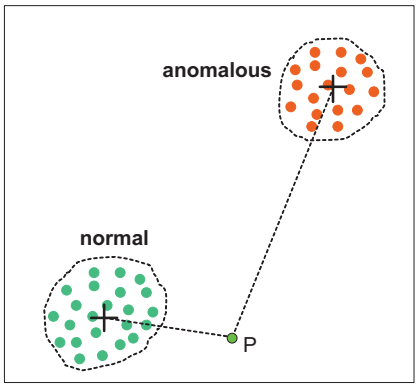


Fig. 2. Classification for $K = 2$

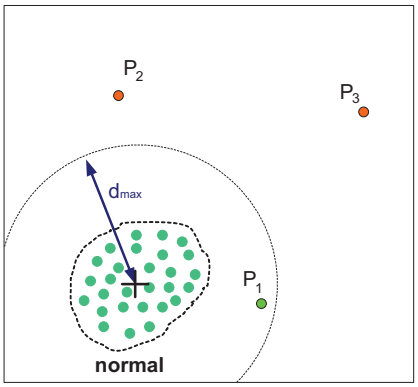


Fig. 3. Outlier detection

similar to normal traffic. Nevertheless, the cluster centroids can still be used for outlier detection as explained in the next subsection.

An essential problem of the K-means clustering method is to define an appropriate number of clusters K. As initial value, we chose $K = 2$, assuming that normal and anomalous traffic in the training data form two different clusters. Obviously, a different number of clusters may result in better clusters, e.g. if the considered service already shows distinct periods of very low and very high traffic volume under normal conditions. However, the determination of an optimum number of clusters based on a cluster evaluation criterion is subject to our ongoing research and not covered in this paper.

C. Classification and Outlier Detection

The K-means clustering process results in cluster centroids for normal and anomalous traffic which can be used to detect anomalies in new flow records monitored in the same network. New flow records have to be preprocessed and transformed like the training data in order to obtain the same features. For the purpose of anomaly detection, we deploy two distance-based methods – classification and outlier detection – that both use the K-means clustering results and that can be applied individually or in a combined way.

Classification. The distances to the cluster centroids of the corresponding traffic class are calculated using the weighted Euclidean distance function. An object is classified as normal if it is closer to the normal cluster centroid than to the anomalous one, and vice versa. This is illustrated in Figure 2 with a two-dimensional feature space: Object P is closer to the normal cluster, therefore P is normal. This distance-based classification allows detecting known kinds of anomalies, i.e. anomalous traffic with similar characteristics as in the training datasets.

Outlier detection. An outlier is an object that differs from most other objects significantly. Therefore it can be considered as an anomaly. For outlier detection, only the distance to the appropriate centroid of the normal cluster is calculated. If the distance between an object and the centroid is larger than a predefined threshold d_{max}, the object is treated as an outlier and anomaly. This is depicted in Figure 3 where P_2 and P_3 lie outside the d_{max} circle. In contrast to the classification method, outlier detection does not make use of the anomalous cluster centroid, i.e. it may be less accurate in detecting known kinds of anomalies. On the other hand, it allows detecting new anomalies that do not appear in the training datasets.

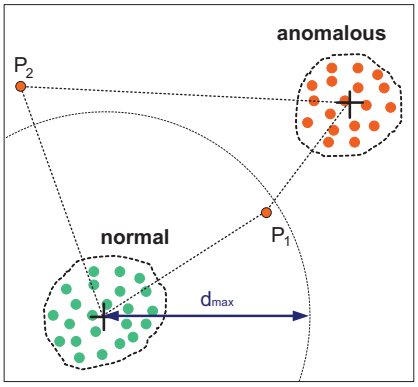


Fig. 4. Combined classification and outlier detection

Combined classification and outlier detection. Classification and outlier detection can be used in a combined way in order to overcome the limitations of each individual method. If the two methods are applied simultaneously, an object is treated as an anomaly if it is closer to the anomalous cluster centroid than to the normal one, or if its distance to the normal cluster centroid is larger than the predefined threshold. In Figure 4, for example, both objects P_1 and P_2 are regarded as anomalies. P_1 is closer to the anomalous cluster and P_2's distance to the normal group is larger than the threshold d_{max}.

In the following section, we present and discuss first experiments with the proposed clustering and anomaly detection approach.

IV. Experimental Results

We tested and evaluated our NDM approach with monitoring data from two different sources. First, we generated traffic in a local testbed. Secondly, we played back tcpdump traces recorded in a real network. Both times, we deployed

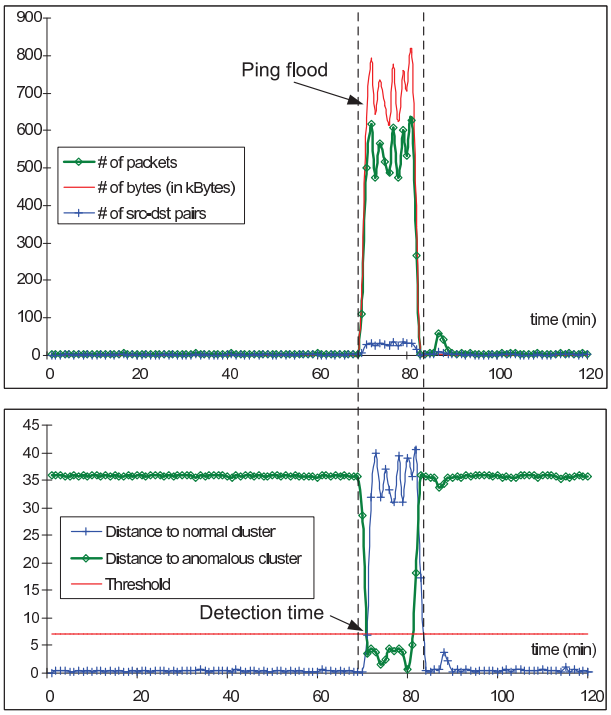


Fig. 5. Ping flood detection with generated traffic

the monitoring probe Vermont [19] to perform flow accounting and to store the resulting flow records in a database. In order to implement the data mining and anomaly detection algorithms, we extended the traffic analysis tool of the HISTORY (HIgh-Speed neTwork mOnitoRing and analYsis) project [20], [21].

These first experiments proof the general feasibility of the concept, yet they are not meant to provide quantitative results, such as false-positive and false-negative ratios. In subsection IV-C, we finally discuss the algorithmic complexity.

A. Evaluation with Generated Traffic

In a testbed environment, we evaluated the capability to detect port scans and DoS attacks. Therefore, we generated normal background traffic and attack traffic using multiple PCs and captured the traffic at the monitoring port of the interconnecting switch. As background traffic, we generated several TCP, UDP and ICMP flows of variable bit rate using the traffic generator *npag* [22]. This tool was also used to produce a ping flood and a UDP flood against port 53 (DNS). Additional port scans were performed with *nmap* [23].

Even though the generated traffic is not representative for real Internet traffic, it enables us to verify the basic capability to detect anomalous attack traffic. As an example, we illustrate the detection of an ICMP ping flood. The K-means clustering algorithm was applied to training data that included flow records of normal and attack traffic. Hence, the data mining process resulted in cluster centroids for normal and anomalous traffic.

Figure 5 shows the detection of a similar ICMP ping flood observed in a monitoring data set. The upper diagram depicts the three features *number of packets*, *number of bytes*, and *number of source-destination pairs* over time for the ICMP traffic class (interval length $T = 1$ min). As can be seen, the ping flood starts at $t = 70$ min. In the lower diagram, it is shown how the anomaly is detected: At $t = 71$ min the distance to the anomalous cluster falls below the distance to the normal cluster. The traffic is classified as normal again after the distance to the normal cluster becomes smaller than the distance to the anomalous cluster and at the same time smaller than the threshold $d_{max} = 7$ used for outlier detection ($t = 84$ min). Similarly, UDP floods as well as port scans could be detected.

B. Evaluation with Real Traces

In order to obtain first results on how the clustering method performs in real networks, we applied it to packet traces recorded at a gateway router that connects a student's residential network to the Internet. These anonymized traces are publicly available as tcpdump files from the traffic measurement data repository of the University of Twente [24], [25]. Each trace file contains a snippet of 15 minutes monitored at a 300Mbit/s Ethernet link.

We played back the trace file loc1-20020526-1115.bz2 at 10 percent of the recording speed using *tcpreplay* [26]. This allowed us to use standard PCs for monitoring and flow accounting without risking packet losses by the software monitoring processes. As a result of the reduced playback speed, the duration of the trace was stretched to 150 minutes. Again, the interval length was set to $T = 1$ min, corresponding to $T = 6$ sec at the original time scale.

A port analysis of the monitoring data revealed that more than 85 percent of both incoming and outgoing traffic was directed to unusual destination port numbers. This corresponds to the expectation that traffic of a student's residential network shows a large amount of traffic from peer-to-peer applications and online games. Nevertheless, we applied the K-means clustering algorithms to specific services such as HTTP, FTP and SSH. The resulting centroids of the normal and anomalous cluster were very close to each other, which indicates that there were no traffic anomalies in the analyzed data. Only the clusters for FTP control traffic (TCP port 21) can be clearly distinguished by a large difference in the number of packets and bytes (the difference in *cn* was small). However, we assume that both clusters represented normal behavior: The cluster with large number of packets and bytes can be explained by FTP clients scanning the directories on an FTP server.

A more interesting result was obtained by analyzing the UDP traffic of the default class. The K-means clustering resulted in the following two cluster centroids:

Cluster	pkts	bytes	src-dst
normal	28274	3288007	1896
anomalous	39725	3510792	14831

As can be seen, the average value of source-destination pairs in

the anomalous cluster is eight times higher than in the normal cluster. The corresponding anomalous traffic was detected in a time interval of about three minutes which corresponds to 15 to 20 seconds on the original time scale. More than 22.000 small UDP packets of 37 bytes were sent from a single IP address in the residential network to more than 20.000 different, mostly external IP addresses. The predominating destination port was 27015 (more than 14.000 packets), the main source port was 1830, followed by 1831, 1832, and 1833. The majority of the transmitted packets resulted in a small reply packet (34 bytes) sent back to the IP address in the residential network. According to some information found on the Web, UDP port 27015 is used by servers of the online game *half-life*. Hence, this traffic anomaly was probably caused by the client program of this game updating its list of available game servers in the Internet.

C. Complexity Estimation

The complexity of the K-means clustering algorithm is $O(K\,n\,t)$ where K is the number of clusters, n the number of objects to be classified, and t the number of iterations which depends on the initial classification of the objects and the feature value distribution (typically, $t << n$). We apply the K-means clustering algorithm with $K = 2$ to the feature datasets of different predefined services separately. If m is the number of time intervals T in the training data, for each traffic class up to m feature datasets may occur. In case that we consider L specific services, we get $O\,(K\,(L+3)m\,t)$ (we have to add 3 for the default traffic classes for TCP, UDP, and ICMP). As a comparison: The LOF clustering algorithm deployed in [7] is $O(n^2)$ complex.

More important than the complexity of the data mining process is the detection complexity that is required to classify new monitoring data as normal or anomalous. This is because a low and constant detection complexity is essential for a scalable real-time detection mechanism. Our distance-based detection requires the transformation of new monitoring data into the feature space. Thereby, the flow records occurring in one time interval T are converted into one feature dataset for each service. The corresponding distances to the normal and anomalous cluster centroids have to be calculated and compared to each other as well as to the outlier detection threshold. Applying the detection to monitoring data of one time interval T is thus $O\,(K(L+3))$ complex.

These estimations show that the complexity of the data mining process is acceptable and that the detection complexity only depends on the number of clusters K and the number of separately considered services L. In order to keep L at a low level without losing the advantage of having distinct profiles for different services, services with similar traffic behavior could be analyzed as a group.

V. Conclusion

In this paper, we presented a novel *Network Data Mining* approach that applies the K-means clustering algorithm to feature datasets extracted from flow records. Training data are divided into clusters of time intervals of normal and anomalous traffic. While the data mining process is relatively complex, the resulting cluster centroids can be used to detect anomalies in new on-line monitoring data with a small number of distance calculations. This allows deploying the detection method for scalable real-time detection, e.g. as part of a intrusion detection system. Applying the clustering algorithm separately for different services (identified by their transport protocol and port number) improves the detection quality. We presented and discussed the results of first experiments using generated and real traffic.

We are currently working on several improvements of the presented approach, thus comparing clustering results achieved with different K in order to determine the optimum number of clusters, considering additional features such as the average flow duration, and considering different distance metrics, such as the Mahalanobis distance. Finally, we are going to evaluate the detection performance using reference data such as DARPA'98 traces [27] as well as real Internet traffic.

References

[1] J. Georges and A. H. Milley, "KDD'99 competition: Knowledge discovery contest," *SIGKDD Explor. Newsl.*, vol. 1, no. 2, pp. 79–84, 2000.

[2] B. Claise, G. Sadasivan, V. Valluri, and M. Djernaes, "Cisco Systems NetFlow Services Export Version 9," RFC 3954 (Informational), Oct. 2004.

[3] B. Claise, S. Bryant, G. Sadasivan, S. Leinen, and T. Dietz, "IPFIX Protocol Specifications," Internet-Draft, work in progress, draft-ietf-ipfix-protocol-24, Nov. 2006.

[4] U. M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, "From data mining to knowledge discovery in databases." *AI Magazine*, vol. 17, no. 3, pp. 37–54, 1996.

[5] W. Lee and S. Stolfo, "Data mining approaches for intrusion detection," in *Proceedings of the 7th USENIX Security Symposium*, San Antonio, TX, 1998.

[6] P. Dokas, L. Ertoz, V. Kumar, A. Lazarevic, J. Srivastava, and P. N. Tan, "Data mining for network intrusion detection," in *Proceedings of the NSF Workshop on Next Generation Data Mining*, Nov. 2002.

[7] L. Ertoz, E. Eilertson, A. Lazarevic, P.-N. Tan, J. Srivastava, V. Kumar, and P. Dokas, *Next Generation Data Mining*. MIT Press, 2004, ch. 3: The MINDS - Minnesota Intrusion Detection System.

[8] M. Esposito, C. Mazzariello, F. Oliviero, S. P. Romano, and C. Sansone, "Evaluating pattern recognition techniques in intrusion detection systems." in *Proceedings of the 5th International Workshop on Pattern Recognition in Information Systems (PRIS) 2005*, May 2005, pp. 144–153.

[9] D. Barbará, J. C. S. Jajodia, L. Popyack, and N. Wu, "ADAM: Detecting intrusions by data mining," in *Proceedings of the IEEE Workshop on Information Assurance and Security*, Jun. 2001, pp. 11–16.

[10] J. Luo and S. Bridges, "Mining fuzzy association rules and fuzzy frequency episodes for intrusion detection," *International Journal of Intelligent Systems*, vol. 15, no. 8, pp. 687–704, 2000.

[11] Libpcap/Tcpdump Homepage, http://www.tcpdump.org, 2007.

[12] W. W. Cohen, "Fast effective rule induction," in *Proceedings of the 12th International Conference on Machine Learning*, A. Prieditis and S. Russell, Eds., Tahoe City, CA, Jul. 1995, pp. 115–123.

[13] R. Agarwal and M. V. Joshi, "PNrule: A new framework for learning classifier models in data mining," Department of Computer Science, University of Minnesota, Tech. Rep. 00-015, 2000.

[14] A. Lazarevic, L. Ertöz, V. Kumar, A. Ozgur, and J. Srivastava, "A comparative study of anomaly detection schemes in network intrusion detection," in *Proceedings of the Third SIAM International Conference on Data Mining*, May 2003.

[15] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "LOF: Identifying density-based local outliers," in *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, May 2000, pp. 93–104.

[16] S. M. Bridges and R. M. Vaughn, "Fuzzy data mining and genetic algorithms applied to intrusion detection," in *Proceedings of the Twenty-third National Information Systems Security Conference*, Oct. 2000.
[17] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability*. University of California Press, 1967, pp. 281–297.
[18] "Introduction to Cisco IOS NetFlow - A Technical Overview," Cisco Systems, Inc., Tech. Rep., Feb. 2006. [Online]. Available: http://www.cisco.com
[19] R. T. Lampert, C. Sommer, G. Münz, and F. Dressler, "Vermont - A Versatile Monitoring Toolkit for IPFIX and PSAMP," in *Proc. of IEEE/IST Workshop on Monitoring, Attack Detection and Mitigation (MonAM 2006)*, Tuebingen, Germany, Sep. 2006.
[20] F. Dressler and G. Carle, "History - high speed network monitoring and analysis," in *Proceedings of the 24th IEEE Conference on Computer Communications (IEEE INFOCOM 2005), Poster Session*, Mar. 2005.
[21] History Project Homepage, http://www.history-project.net, 2007.
[22] F. Dressler, "Policy-based traffic generation for IP-based networks," in *25th IEEE Conference on Computer Communications (IEEE INFOCOM 2006), Poster Session*, Apr. 2006.
[23] Nmap Network Scanner Homepage, http://insecure.org/nmap/, 2007.
[24] R. van de Meent, "M2C Measurement Data Repository," University of Twente, Enschede, The Netherlands, M2C Deliverable D1.5, Dec. 2003.
[25] University of Twente - Traffic Measurement Data Repository, http://traffic-repository.ewi.utwente.nl, 2007.
[26] Tcpreplay Homepage, http://tcpreplay.synfin.net/trac/, 2007.
[27] R. P. Lippmann, D. J. Fried, I. Graf, J. W. Haines, K. R. Kendall, D. McClung, D. Weber, S. E. Webster, D. Wyschogrod, R. K. Cunningham, and M. A. Zissman, "Evaluating Intrusion Detection Systems: the 1998 DARPA Off-Line Intrusion Detection Evaluation," in *Proceedings of the 2000 DARPA Information Survivability Conference and Exposition*, 2000.

Konzeption und Entwicklung eines echtzeitfähigen Lastgenerators für Multimedia-Verkehrsströme in IP-basierten Rechnernetzen

Andrey W. Kolesnikov

Universität Hamburg, Department Informatik
Telekommunikation und Rechnernetze (TKRN)
6kolesni@informatik.uni-hamburg.de

Abstract: Experimente zur Leistungsanalyse von dienstintegrierten Rechnernetzen sind stets unter Last durchzuführen. Der Einsatz von künstlichen (synthetischen) Lasten bringt hier signifikante Vorteile. Demzufolge wächst auch der Bedarf nach entsprechenden geeigneten Spezialwerkzeugen zur Lastgenerierung in existierenden Netzen. In dem vorliegenden Beitrag wird eine Architektur für einen echtzeitfähigen Lastgenerator basierend auf einer dedizierten Lastspezifikationstechnik skizziert und einige zu berücksichtigende Realisierungsaspekte aufgezeigt. Ausgehend von einem existierenden Prototyp eines Lastgenerators (UniLoG) soll eine echtzeitfähige Version des bestehenden Lastgenerierungswerkzeuges mit möglichst breitem Anwendungsspektrum erstellt werden.

1. Motivation und Zielsetzung

Analysen und Prognosen der Leistungsfähigkeit eines Kommunikationssystems werden üblicherweise unter verschiedenen Belastungsniveaus durchgeführt. Der Einsatz von künstlichen Lasten besitzt hierbei gegenüber den Lasten aus einer realen Anwendung die Vorteile der Reproduzierbarkeit, Skalierbarkeit und Flexibilität in der Parametrisierung. Somit steigt der Bedarf nach speziellen Werkzeugen zur universellen Lastbeschreibung und flexiblen Lastgenerierung an verschiedenen Schnittstellen in realen und emulierten Netzen [Sch06, SBW05].

Eine in der aktuellen Literatur gängige Methode zur Erzeugung von realitätsnahen künstlichen Lasten besteht in der Realisierung eines Lastgenerators in Form von einem Lastmodell, welches aus umfangreichen Messungen an einem realen Netz abgeleitet wurde. Viele verschiedene modell-basierte Lastgeneratoren zur Erzeugung von realitätsnahen Verkehrsströmen wurden bereits vorgeschlagen [BaC98], [BGP05], [BuW02], [Hee00], [MoG03], [RRB07], [SoB04], [TFCV03]. Die existierenden Lösungen sind allerdings durch einige wesentliche Einschränkungen und Nachteile gekennzeichnet, wie z.B.:

- geringe Flexibilität bei der Erzeugung verschiedener Lastprofile (resultierend im Wesentlichen aus der starken Orientierung an einem konkreten Modellierungsfall),
- fehlende Unterstützung einer formalen Lastspezifikationstechnik und einer Funktion zur Erzeugung von Lastmodellen (bestehende Lösungen sind häufig nur Programme, die lediglich ein bestimmtes Modell implementieren),
- keine saubere Trennung zwischen der Lastbeschreibung (im Sinne einer Spezifikation abstrakter Aufträge) und der Lastgenerierung,
- keine Möglichkeit zur Anpassung an die verschiedenen Netzschnittstellen und Vernachlässigung der Abhängigkeit des Benutzerverhaltens von dem Netzzustand bei der Lastmodellierung.

Aus diesen Gründen wurde an der Arbeitsgruppe TKRN ein Konzept für den Entwurf eines verallgemeinerten Lastgenerators UniLoG entwickelt ([CoK05], [Con06]), in dem

- verschiedene Lastmodelle mithilfe einer formalen automatenbasierten Lastspezifikationstechnik von den Benutzern des Lastgenerators erstellt werden können,
- eine strukturierte Methode zur Lastspezifikation an verschiedenen Netzschnittstellen unterstützt wird,
- die Lastparameter gemäß realen Traces oder analytischen Modellen konfiguriert werden können,
- dynamische Lastgenerierung durch Berücksichtigung von netzbedingten Wartezuständen der Benutzer ermöglicht wird.

Die vorliegende Arbeit ist in den Kontext der laufenden Arbeiten der Arbeitsgruppe TKRN im Bereich Traffic-Engineering eingebettet. Auf der Basis einer automatenbasierten Lastbeschreibungstechnik [Wol99], [Bai99], [CoW06] soll eine in [Con06] entworfene Architektur zur Lastgenerierung in dienstintegrierten Netzen prototypisch realisiert werden, die die Generierung von Multimedia-Verkehrsströmen in Echtzeit ermöglicht. Bei der Modellierung des Verhaltens der auftragsgenerierenden Benutzer soll die Möglichkeit zur Blockierung durch netzbedingte Wartezustände berücksichtigt werden.

Als besondere Randbedingungen werden die Anforderungen an die Leistungsfähigkeit und Präzision bei der Auslieferung zu generierender Aufträge betrachtet. Die Möglichkeit zur Überlagerung mehrerer Verkehrsströme wird als wünschenswerte Erweiterung des geplanten Entwurfs angestrebt.

Im folgenden Abschnitt werden die Grundlagen der Lastgenerierung, insbesondere die verwendete automatenbasierte Lastspezifikationstechnik, kurz zusammengefasst. Die auf dieser Lastspezifikationstechnik basierende bestehende Architektur des Lastgenerators UniLoG wird in Abschnitt 3 als Ausgangssituation dargestellt und im Hinblick auf die für eine echtzeitfähige Version notwendigen Erweiterungen betrachtet. In Abschnitt 4 werden zunächst die kritischen Probleme und Einflussfaktoren einer Echtzeitumgebung aufgezeigt und anschließend die entsprechenden Mittel und Maßnahmen zu ihrer Lösung in dem Entwurf eines erweiterten Lastgenerators UniLoG vorgeschlagen. Die wichtigsten Aspekte der Realisierung von Teilkomponenten des neuen Entwurfs, insbesondere die realisierte Synchronisationsfunktion zur sicheren Kontrollübergabe, werden in Abschnitt 5 erläutert. In Abschnitt 6 werden die ersten Analysen der Präzision und der Leistungsfähigkeit der realisierten Lösung präsentiert. Eine Zusammenfassung der erzielten Ergebnisse und Möglichkeiten für weitergehende Forschungsaktivitäten schließt diesen Artikel ab.

2. Grundlagen der Lastgenerierung

Der Einsatz von künstlichen (synthetischen) Lasten beim Traffic-Engineering von Netzen setzt eine geeignete Lastspezifikationstechnik voraus. Eine solche Methode zur formalen Lastbeschreibung wurde an der Arbeitsgruppe TKRN erarbeitet [Wol99], [Bai99], aktuell erweitert in [CoW06] und basiert im Wesentlichen auf folgenden vier Schritten:

(1) Festlegung der Schnittstelle *IF*, an der die Auftragsgenerierung stattfinden soll (z.B. Transportdienstschnittstelle); dadurch wird die Trennung der lastgenerierenden Umgebung *E* von dem auftragsbearbeitenden System *S* vorgenommen.
(2) Spezifikation der möglichen abstrakten Auftragstypen und ihrer Attribute.
(3) Spezifikation der möglichen Auftragssequenzen durch einen zustandsbasierten Benutzerverhaltensautomaten (*BVA*).
(4) Spezifikation der Attributwerte und der Übergabezeitpunkte der abstrakten Aufträge an *IF*.

Eine solche Technik erlaubt es, die an der Schnittstelle *IF* in einem Zeitintervall *T* angebotene Last L_T als Sequenz von Aufträgen $L_T=(t_i, A_i), 0 \leqslant t_i \leqslant T, i=1,2,...,n$ zu beschreiben (t_i: Übergabezeitpunkt, A_i: der Auftrag mit seinen Attributen).

Die transformierende Wirkung von Teilen des Bediensystems *S* (z.B. Header-Generierung oder Fragmentierung der Aufträge) kann bei diesem Konzept in einer Komponente *Lasttransformator* modelliert werden.

Die vorgeschlagene formale Lastspezifikationstechnik verwendet einen *Benutzerverhaltensautomaten* (BVA) $U=\{\varphi_i, \varphi_a, \varphi_b, \varphi_t, T_\varphi\}$ zur Beschreibung des Lastgenerierungsprozesses. Dabei sind φ_i, φ_a, φ_b, φ_t die sog. Makrozustände des BVA und T_φ ist die Menge der Transitionen (Übergänge) zwischen diesen Makrozuständen (kurz *M-Zuständen*, vgl. Abb. 2.1). Die einzelnen *M-Zustände* sind wie folgt definiert:

- φ_i: *der Initialisierungszustand*, in dem der Benutzer vor dem Start der Lastgenerierung residiert. Der Lastgenerierungsprozess wird gestartet durch Verlassen von φ_i zu einem bestimmten Zeitpunkt τ_0.
- φ_a : *aktiver M-Zustand*, bestehend aus einer Menge von auftragserzeugenden Zuständen (*R-Zuständen*) und Verzögerungszuständen (*D-Zuständen*).
- φ_b : *blockierter M-Zustand*, in dem das Warten des Benutzers auf Systemreaktionen explizit modelliert wird.
- φ_t : *terminierter M-Zustand*, der Lastgenerierungsprozess endet stets in diesem Zustand (wenn der BVA in einem Lastgenerator eingesetzt wird, wird der entsprechende Lastgenerator beim Erreichen von φ_t die weitere Auftragserzeugung sofort beenden).

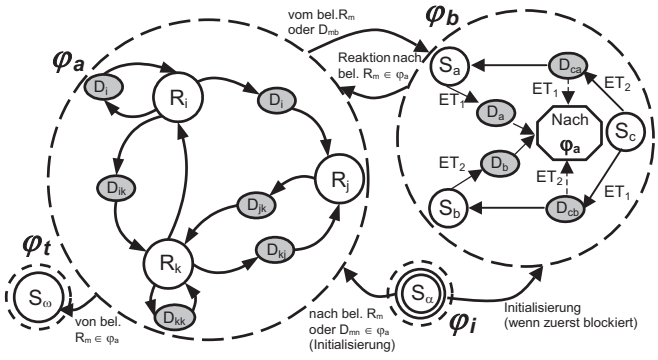


Abbildung 2.1: Eine exemplarische Verfeinerung der Makrozustände eines BVA

Jeder Makrozustand besteht aus der Menge weiterer *LG-Zustände* (oder kurz *Zustände*), in denen die Spezifikation des Benutzerverhaltens weiter verfeinert wird[1]:

- *S-Zustände*: hier muss ein Ereignis abgewartet werden, bevor der Zustand verlassen werden kann. Mögliche Ereignisse sind die *Initialisierung* (im

[1] Zur detaillierten Beschreibung der Komponenten eines BVA sowie der Möglichkeiten für ihre Parametrisierung verweisen wir an dieser Stelle auf [CoW06].

einzigen Zustand S_α des M-Zustands φ_i) und die *Termination* (im einzigen Zustand S_ω des M-Zustands φ_t) des Lastgenerierungsprozesses sowie die *Reaktionen*, die in dem blockierten M-Zustand φ_b modelliert und interpretiert werden. Mögliche Reaktionen werden in Reaktionstypen ET_1, ET_2, ..., ET_n klassifiziert.

- *R-Zustände*: hier werden Aufträge exakt eines bestimmten Typs generiert. Nach der Generierung des nächsten Auftrags wird der Zustand sofort verlassen. R-Zustände sind nur in dem aktiven M-Zustand φ_a enthalten.
- *D-Zustände*: hier werden die Verzögerungen vor der Generierung des nächsten Auftrags (die Zwischenankunftszeiten der Aufträge) in φ_a bzw. die Dauer der Verarbeitung und der Interpretation einer Systemreaktion in φ_b modelliert.

Die Möglichkeiten zur Parametrisierung der Komponenten des BVA zur Generierung von synthetischen Lasten sind gegeben

- bei Zustandsübergängen (Transitionen): durch Traces, Übergangswahrscheinlichkeiten oder spezielle Prozeduren,
- bei Verzögerungen in den D-Zuständen und Auftragsattributen: durch konstante Werte, Traces, Wahrscheinlichkeitsverteilungen und spezielle Prozeduren.

Die Eingabedaten für den BVA sind die externen Ereignisse, welche die Reaktionen des Systems an den BVA signalisieren. Die Ausgabedaten des BVA liegen als eine Sequenz von Aufträgen vor, die in den R-Zuständen generiert werden.

3. Ausgangssituation und Randbedingungen dieser Studie

Basierend auf der beschriebenen Lastspezifikationstechnik werden in [CoK05] folgende Schritte für die Modellierung des Lastgenerierungsprozesses mithilfe eines verallgemeinerten Lastgenerators vorgeschlagen:

Schritt 1: Erstellen von Lastmodellen an einer wohldefinierten Schnittstelle in Form eines BVA und Konfigurierung der entsprechenden Modellparameter in einem parametrisierten BVA (PBVA).

Schritt 2: Generierung von abstrakten Aufträgen gemäß den Parametern des Lastmodells und Übergabe der Aufträge entsprechend den spezifizierten Übergabezeitpunkten an die auftragsausführenden Komponenten.

Schritt 3: Transformation der abstrakten Aufträge in Aufträge an einer weiter unten im Protokollstapel liegenden Schnittstelle, falls das verwendete Lastmodell nicht dem Lastmodell entspricht, welches zur Generierung von Aufträgen an der vom Experimentator gewählten Schnittstelle benötigt wird.

Schritt 4: Ausführen von Aufträgen und Übergabe der realen Last an das Netz.

Eine entsprechende Architektur zur Lastgenerierung wurde in [CoK05] und [Con06] vorgeschlagen (siehe Abb. 3.1). In dem Entwurf wird der grundlegende Gedanke der Trennung zwischen der Lastspezifikation (resultierend in den abstrakten Aufträgen) und der Lastgenerierung (resultierend in den realen bzw. konkreten Aufträgen an der vom Experimentator gewählten Schnittstelle *IF*) befolgt.

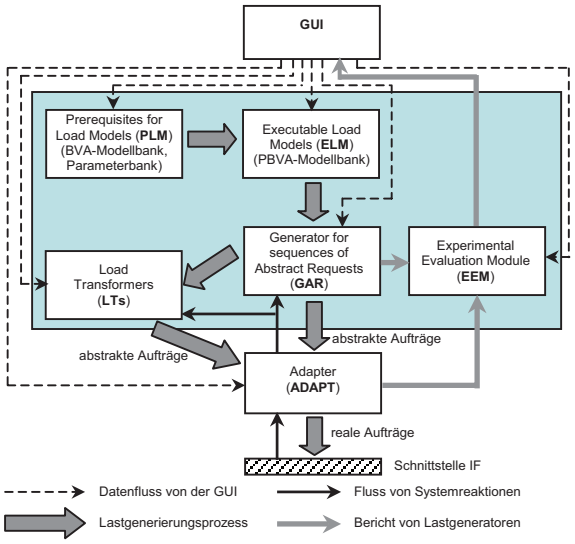


Abbildung 3.1: Hauptkomponenten eines verallgemeinerten Lastgenerators UniLoG (Grobsicht der Architektur)

Das prototypisch implementierte Werkzeug zur Lastgenerierung UniLoG beinhaltet Komponenten zur graphischen Spezifikation des Benutzermodells (*BVA*), dessen Parametrisierung (*PBVA*) sowie zur Generierung von zunächst abstrakten Aufträgen (*GAR*) und ermöglicht gegenwärtig bereits die Generierung von abstrakten Aufträgen anhand von elementaren Modellen ohne Überlagerung und ohne netzbedingte Blockierungszustände der Benutzer (z.B. Übertragung eines PCM-codierten Audiostroms oder einer MPEG-codierten Videosequenz).

Mithilfe einer schnittstellen-spezifischen Adapterkomponente *ADAPT* können aus den abstrakten Aufträgen reale (um die schnittstellen-spezifischen Attribute ergänzte) Aufträge an der gewählten Schnittstelle *IF* generiert werden (siehe Abb. 3.1). Der prototypisch realisierte Lastgenerator lieferte für zahlreiche Testszenarien bereits sehr gute Ergebnisse bezüglich der Präzision bei der Übergabe generierter Aufträge [CoK05].

Darüber hinaus wurden im Rahmen des Projektes TeleMuM an der AG TKRN die Lernwerkzeuge *LoadSpec* und *LoadTrafo* entwickelt mit dem Ziel, die Verfahren und Techniken der Lastbeschreibung mithilfe der Benutzerverhaltensautomaten sowie

einer Lasttransformation für die Studierenden zu veranschaulichen [FHW04].

Basierend auf den Erkenntnissen aus den Experimenten mit UniLoG [CoK05] entstand konsequenterweise eine Idee für den Entwurf eines echtzeitfähigen Lastgenerators. Eine besondere Herausforderung an den neuen, erweiterten Lastgenerator stellt die Realisierung der netzbedingten Blockierungszustände der Benutzer dar (die z.B. zur Modellierung der Flusskontrolle bei TCP benötigt werden). Die in dem Adapter erforderliche Erfassung und Verarbeitung der entsprechenden Systemreaktionen und ihre Übergabe an den Generator als abstrakte Reaktionsnachrichten (z.B. für die Wahl des Nachfolgers des aktuellen S-Zustandes innerhalb von φ_b im BVA) lässt die bei der Entwicklung von UniLoG gemachte Annahme über die zeitlich getrennte Generierung der abstrakten Aufträge (im Generator) und der realen Aufträge (im Adapter) aufheben.

4. Entwurf für einen echtzeitfähigen Lastgenerator

In der neuen Architektur müssen das Benutzermodell (parametrisierter *BVA*) im Generator und der Adapter nebenläufig ausgeführt werden. Darüber hinaus wäre es wünschenswert, Auftragsströme mehrerer modellierter Benutzer (*BVA*) in Echtzeit zu überlagern, um ein spezielles Traffic-Mix zu erzeugen (z.B. Übertragung eines MPEG-Videostroms über UDP mit Dateitransfer über TCP als Hintergrundlast). Die Überlagerung bzw. die Superposition der entsprechenden Auftragssequenzen kann hierbei als ein Spezialfall der Lasttransformation angesehen werden [Bai99].

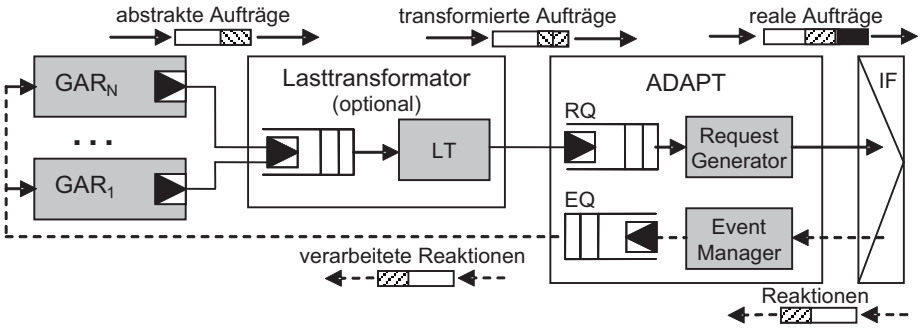


Abbildung 4.1: Architektur eines echtzeitfähigen Lastgenerators

Die bereits bestehenden Komponenten zur Modellspezifikation (*BVA*), Parametrisierung (*PBVA*) und statistischen Auswertung der Experimentergebnisse (*EEM*) sollen weiter genutzt werden. Sie sind in ihrer Funktionsweise von den geplanten Änderungen nicht generell betroffen und können nach geringfügigen Anpassungen in den neuen Entwurf übernommen werden.

Auf der Basis dieser Überlegungen wurde im Kern des neuen Lastgenerators die Aufteilung in den Generator abstrakter Aufträge (*GAR*), den Lasttransformator (*LT*) und den schnittstellenspezifischen Adapter (*ADAPT*) vorgenommen (siehe Abbildung 4.1).

Zwischen dem Generator abstrakter Aufträge und dem Adapter besteht eine bidirektionale Produzenten-Konsumenten Beziehung. Einerseits produziert der Generator abstrakte Aufträge in den R-Zuständen des BVA und stellt sie (ggf. nach einer Transformation in *LT*) in die Auftragswarteschlange RQ des Adapters ein. Diese Aufträge werden vom Adapter konsumiert, indem sie um die schnittstellenspezifischen Attribute bzw. um die fehlenden Attributwerte ergänzt und als reale Aufträge zum spezifizierten Übergabezeitpunkt an IF ausgeliefert werden.

Andererseits verarbeitet der Adapter die Systemreaktionen an *IF* und produziert entsprechende abstrakte Reaktionsnachrichten in der Reaktionswarteschlange EQ. Diese Reaktionsnachrichten werden von dem Generator konsumiert, indem z.B. der Nachfolger des aktuellen Blockierungszustandes im BVA ermittelt wird.

Besondere Randbedingungen für den Entwurf der entsprechenden Teilkomponenten für den Generator und den Adapter bilden folgende Umstände:

(1) die Warteschlangen RQ und EQ haben eine beschränkte Kapazität
(2) Prozesse sollen nur dann aktiv sein, wenn die zu bedienende Warteschlange nichtleer ist
(3) alle Aufträge sind zeitkritisch (die spezifizierten Übergabezeitpunkte sind einzuhalten).

Da die Warteschlangen in diesem Entwurf im gemeinsam genutzten Speicher liegen (alternativ könnte das sog. "Message-Passing" als Methode zur Interprozesskommunikation zwischen dem Generator und dem Adapter eingesetzt werden), soll zur Lösung der Probleme (1) und (2) zunächst der wechselseitige Ausschluss für den Generator und den

Adapter beim Zugriff auf RQ und EQ realisiert werden. Darüber hinaus ist eine explizite Synchronisation zwischen dem Generator und dem Adapter erforderlich, um z.B. das Einfügen eines Auftrags in die volle RQ bzw. das Entnehmen eines Auftrags aus der leeren RQ zu verhindern (das Gleiche gilt für die Reaktionen in der EQ).

Die Einhaltung der Echtzeitanforderungen (3) in Bezug auf die Auslieferung der realen Aufträge zu präzise spezifizierten Übergabezeitpunkten wird insbesondere durch die folgenden Faktoren erschwert:

- modellbedingte Faktoren: alle an der Lastgenerierung beteiligten Komponenten und Prozesse benötigen zur Bearbeitung des nächsten generierten Auftrages eine bestimmte Verarbeitungszeit
- betriebssystembedingte Faktoren (Einfluss anderer Prozesse im Multitasking-Betrieb)
- Besonderheiten zu generierender Auftragssequenzen, z.B. stoßweises Auftragsaufkommen ("Bursts").

Die endlichen Verarbeitungszeiten der an der Auftragsgenerierung beteiligten Komponenten lassen sich generell nicht beseitigen. Die einzigen Möglichkeiten, die Verarbeitungszeiten zu verringern, sind die Optimierung der Modellausführung (in dem Generator bzw. in dem Transformator), der Einsatz einer leistungsfähigeren Hardware bzw. Parallelisierung der Modellausführung auf mehrere Maschinen. Alternativ wäre die Vorbereitung einer bestimmten Anzahl von Aufträgen im Voraus sinnvoll, soweit die Abhängigkeit des Benutzerverhaltens von den Systemreaktionen dies ermöglicht.

Die betriebssystembedingten Faktoren resultieren i.d.R. aus dem Multitasking-Betrieb. Der Ablaufplaner des Betriebssystems kann z.B. einen anderen Prozess aktivieren, wenn in dem Adapter gerade kritische Aufträge zur Versendung vorliegen. Die sicherste Methode hier wäre die Verwendung eines Echtzeit-Betriebssystems wie z.B. WindowsCE [MSCE] bzw. RTLinux [RTL]. Weitere Möglichkeiten sind die Verwendung einer dedizierten Station zur Lastgenerierung und gezieltes Abschalten störender (Hintergrund-)Prozesse.

In dieser Arbeit wurde der Lastgenerator zunächst auf der Basis eines nicht echtzeitfähigen Betriebssystems entwickelt, wobei ein Übergang zu einem Echtzeitbetriebssystem später ohne grundlegende Änderungen der Architektur vorgenommen werden kann. Hierbei wurde besonders auf gezielte Abschaltung störender Prozesse sowie auf die Vergabe von entsprechenden Prioritäten für die einzelnen Komponenten der Architektur geachtet. Leistungsanalysen des ersten realisierten Prototypen brachten bereits sehr ermutigende Ergebnisse in Bezug auf die Einhaltung der spezifizierten Übergabezeitpunkte der Aufträge auch bei Einsatz eines nicht-echtzeitfähigen Betriebssystems.

Besonderheiten in der zu generierenden Auftragssequenz, insbesondere die sog. Auftragsbursts, können letztlich mithilfe der Beobachtung von Dringlichkeit der abstrakten Aufträge und gezielter vorzeitiger Aktivierung des Adapters bewältigt werden, soweit die Auftragszwischenankunftszeiten die maximal benötigte Verarbeitungszeit T_{ADAPT} in dem Adapter nicht unterschreiten[2]. Die realisierte Synchronisationsmethode zur sicheren Kontrollübergabe zwischen dem Generator und dem Adapter wird im Abschnitt 4.3 behandelt.

4.1 Generator abstrakter Aufträge

Die Rolle des GAR-Generators im Lastgenerierungsprozess ist die Erzeugung einer Sequenz von abstrakten Aufträgen (t_i, r_i) durch die Traversierung des entsprechenden PBVA (dabei ist t_i der physikalische Übergabezeitpunkt des Auftrags r_i). Generell unterscheiden wir zwischen den *logischen* (lokalen, relativen) Übergabezeitpunkten in dem BVA-Modell und den *physikalischen* (globalen, absoluten) Übergabezeitpunkten in der resultierenden Auftragssequenz, die Konvertierung dieser Zeitpunkte nimmt der Generator bei der Erzeugung des nächsten Auftrages vor. Im Initialisierungszustand φ_i des PBVA ist die logische Zeit im Generator gleich Null. Der Lastgenerierungsprozess beginnt mit dem Verlassen des Initialisierungszustandes φ_i zum physikalischen Zeitpunkt ι_0. Der physikalische Übergabezeitpunkt t_i des nächsten Auftrags ergibt sich aus dem spezifizierten logischen Übergabezeitpunkt im PBVA plus die physikalische Zeit τ_0, zu der der Lastgenerator gestartet wurde (siehe Abb. 4.2).

Während der Traversierung des PBVA werden in den R-Zuständen abstrakte Aufträge als Objekte mit ihren Attributen erzeugt, mit dem entsprechenden physikalischen Übergabezeitpunkt versehen und am Ende der Auftragswarteschlange *RQ* eingestellt, so dass der nächste zu versendende Auftrag stets im Kopf der *RQ* steht.

In der echtzeitfähigen Version bereitet der Generator abstrakte Aufträge im Voraus vor, solange er nicht in den blockierten Makrozustand φ_b des PBVA geht, nicht terminiert ist und die Warteschlange RQ nicht voll wird. Eine bedingungslose Übergabe der Kontrolle an den Adapter nach der Generierung jedes abstrakten Auftrags wäre hier sicher ineffizient, da der physikalische Übergabezeitpunkt (also die Dringlichkeit des abstrakten Auftrags) nicht berücksichtigt bliebe.

[2] Bei Einsatz eines Lasttransformators dürfen die Zwischenankunftszeiten der abstrakten Aufträge auch die maximale Auftragsbearbeitungszeit des Transformators T_{LT} nicht unterschreiten. Während dies bei analytischen Transformatoren relativ unkritisch ist, kann diese Bedingung insbesondere bei simulativen Transformatoren häufig verletzt sein.

Andererseits ist es unser Ziel, die abstrakten Aufträge möglichst frühzeitig an den Adapter zu übergeben. Aus diesem Grund erlauben wir zunächst, dass die logische Uhr in dem Generator der physikalischen Uhr in dem Adapter (die der realen Systemzeit entspricht) vorgehen darf. Der Generator kann z.B. in 500µs Realzeit abstrakte Aufträge für 1 sek. logischer Zeit erzeugen und danach in den blockierten Zustand gehen, in dem die logische Uhr von der physikalischen Uhr bei Eintritt des erwarteten Ereignisses wieder eingeholt wird (s. e^*_1 in Abb. 4.2).

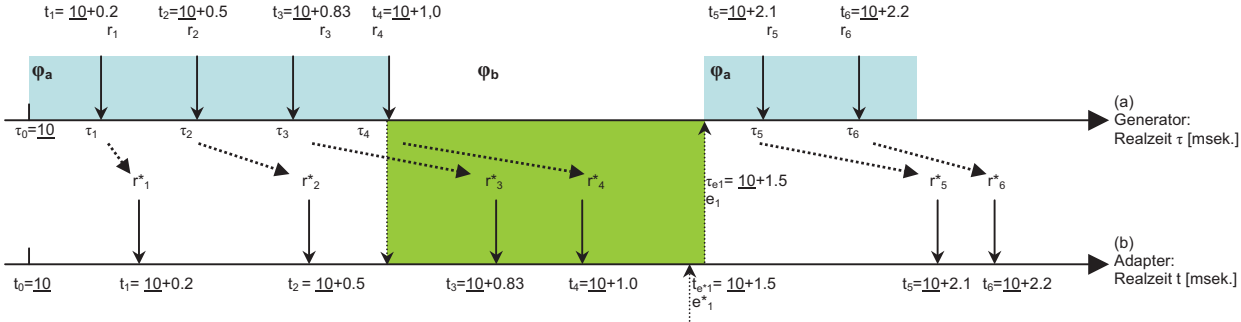


Abbildung 4.2 LG gestartet um $\tau_0 = 10$ Uhr physikalischer Zeit; (a) Abstrakte Aufträge (t_i, r_i) mit physikalischen Übergabezeitpunkten t_i generiert vom Generator zu den realen Zeitpunkten τ_i; (b) Reale Aufträge (t_i, r^*_i) generiert vom Adapter zu den realen Zeitpunkten t_i.

In dem blockierten Makrozustand φ_b des PBVA ist der Generator *konzeptionell passiv* (d.h. der Lastgenerierungsprozess befindet sich in einem blockierten Zustand in Erwartung bestimmter oder unerwarteter Systemreaktionen), *programmtechnisch* kann er jedoch sowohl *aktiv* als auch *passiv* realisiert werden. Die programmtechnisch passive Realisierung wäre vorteilhaft, da in diesem Fall das aktive Warten des Generators auf die Reaktionsnachrichten in der *EQ* vermieden wird.

Wenn eine Systemreaktion vorliegt, kann der Adapter den Generator gezielt aktivieren und über die Systemreaktion mit einer entsprechenden Nachricht in der *EQ* informieren. Der Generator nimmt anschließend folgende Aktionen vor:

(1) die Auswahl des nächsten Zustandes in dem PBVA abhängig vom Typ der Reaktion
(2) Fortschalten der logischen Uhr auf den Zeitpunkt der Entblockierung, der vom Adapter in der Reaktionsnachricht übermittelt wurde und
(3) Verlassen des blockierten Makrozustands φ_b des PBVA.

Nach diesen Aktionen befindet sich der Generator wieder in dem aktiven Makrozustand φ_a und der Lastgenerierungsprozess kann fortgesetzt werden.

4.2 Der Adapter

Die Rolle des schnittstellenspezifischen Adapters im Lastgenerierungsprozess ist

(a) die Konvertierung der abstrakten Aufträge in der *RQ* in reale Aufträge an *IF* sowie
(b) Modellierung von Systemreaktionen an *IF* und Generierung von entsprechenden abstrakten Reaktionsnachrichten in der *EQ*.

Reale Aufträge (t_i, r^*_i) erzeugt der Adapter durch den Aufruf entsprechender Dienstprimitiven (i.d.R. als API-Funktionen vorhanden) an der Schnittstelle *IF* zu den physikalischen Übergabezeitpunkten t_i, die vom Generator in den abstrakten Aufträgen (t_i,r_i) eingetragen wurden (siehe Abb. 4.2). Dabei wendet der Adapter die Konvertierungsvorschriften an, die von dem Experimentator während der Spezifikation des BVA für die abstrakten Auftragstypen und Auftragsattribute eingetragen wurden. Die in der BVA-Spezifikation fehlenden Attribute, die Pflichtparameter der Dienstprimitiven an der *IF* sind, werden mit Standardwerten aus dem PBVA belegt.

Für die Modellierung von Systemreaktionen gibt es grundsätzlich mehrere Möglichkeiten:

(1) *Lokales Modell*: Generierung von abstrakten Reaktionsnachrichten (und ggf. Emulation von netzbedingten Verzögerungen) gemäß den Spezifikationen in dem blockierten Makrozustand φ_b des PBVA.

(2) *Globales (externes) Modell*: Generierung von abstrakten Reaktionsnachrichten gemäß einem analytischen oder simulativen Modell des Netzverhaltens, welches in einem Netzemulator ausgeführt wird [Sch06]. Zur Parametrisierung des Netzmodells (bei simulativer Auswertung) können z.B. lastabhängige Traces verwendet werden, die aus den Messungen an einem realen Kommunikationssystem gewonnen wurden [Kam04].

(3) *Hybride Methode*: Verarbeitung von realen Reaktionen an Schnittstelle *IF* (z.B. ICMP-Nachrichten "Source Quench", "Destination Unreachable", "Time Exceeded" etc.). Dabei wird insbesondere die Verfügbarkeit der entsprechenden Dienstprimitiven bzw. Funktionscodes oder Rückgabewerte zum Abruf von Statusinformationen des Systems vorausgesetzt.

Alle Methoden setzen eine formale Spezifikation der möglichen Reaktionstypen an *IF* voraus, die von dem Experimentator während der Erstellung des BVA vorzunehmen ist. Im Fall (3) sind die Att-

ribute der realen Reaktion auf die Attribute der abstrakten Reaktionsnachricht abzubilden.

In unserem Entwurf werden die Reaktionsnachrichten zunächst gemäß einem lokalen Modell (gemäß den Spezifikationen in dem blockierten Makrozustand φ_b des PBVA) generiert. Die Auslagerung der Modellierung von netzabhängigen Aktivitäten des Benutzers in eine separate Adapter-Komponente ermöglicht später die Einbindung von externen Modellen des Netzverhaltens.

Wie bereits erwähnt, gibt der Generator die abstrakten Aufträge (t_i, r_i) möglichst frühzeitig (also vor ihrem physikalischen Übergabezeitpunkt t_i) an den Adapter weiter, um die Verarbeitungszeit $T_{max,ADAPT}$ in dem Adapter auszugleichen. Der Adapter bereitet zunächst den realen Auftrag (t_i, r^*_i) vor und wartet dann auf seinen physikalischen Übergabezeitpunkt t_i. Um diese aktiven Wartephasen des Adapters möglichst kurz, d.h. im Bereich weniger µs, zu halten bzw. gar zu vermeiden, wurde eine Funktion zur sicheren Kontrollübergabe zwischen dem Generator und dem Adapter entwickelt, auf die im nächsten Abschnitt 4.3 eingegangen wird.

4.3 Sichere Kontrollübergabe zwischen dem Generator und dem Adapter

Wie bereits erläutert, besteht zwischen dem Generator und dem Adapter eine beidseitige "Produzenten-Konsumenten"-Beziehung mit der Interprozesskommunikation über den gemeinsam genutzten Speicher (Warteschlangen *RQ* und *EQ*). In der Lösung des klassischen "Produzenten-Konsumenten"-Problems mit Semaphoren (siehe z.B. [Tan03], [Ste99]) ist der Konsument anfänglich blockiert (*RQ* leer). Der Produzent fängt an, Aufträge zu erzeugen, und blockiert, wenn *RQ* voll wird. Dabei kann (muss aber nicht!) der Scheduler des Betriebssystems die Kontrolle übergeben:

- vom Produzenten (Generator) an den Konsumenten (Adapter) nach dem Einstellen des nächsten Auftrags in die *RQ* bzw.
- vom Konsumenten (Adapter) an den Produzenten (Generator) nach dem Herausnehmen und Bearbeiten des Auftrags aus der *RQ*.

Unter Zugrundelegung einer fairen Bedienstrategie (z.B. Round Robin) würde also eine Abwechslung des Produzenten und des Konsumenten beim Zugriff auf die Warteschlange stattfinden. Eine solche Abwechselung (Alternation) des Generators und des Adapters beim Zugriff auf die *RQ* wäre in unserem Fall zeitkritischer Aufträge insbesondere bei Unregelmäßigkeiten in der zu generierenden Auftragssequenz und steigenden Verarbeitungszeiten in dem Generator und ggf. dem Transformator (hervorgerufen z.B. durch den Einsatz komplexerer Simulationsmodelle) nicht effizient, da der Übergabezeitpunkt (die *Dringlichkeit*) der Aufträge nicht berücksichtigt bliebe.

Der Scheduler des Betriebssystems könnte z.B. den Generator genau dann aktivieren, wenn Aufträge zur Versendung vorliegen und der Adapter aktiviert werden sollte. Allein durch die Vergabe entsprechender Prioritäten für den Generator und den Adapter lässt sich das Problem nicht lösen. An dieser Stelle wird eine Synchronisationsmethode benötigt, bei der die Entscheidung über die Aktivierung des Generators oder des Adapters *anhand der Dringlichkeit der abstrakten Aufträge* getroffen wird.

Wir nennen einen Auftrag A *dringend (urgent)* zum Zeitpunkt t_{NOW}, wenn bis zu seinem physikalischen Übergabezeitpunkt t_{NEXT} weniger als eine festgelegte Zeit Δt übrig bleibt:

$$(t_{NEXT} - t_{NOW}) < \Delta t$$

Die Hauptidee besteht darin, den Generator nach der Erzeugung jedes nächsten abstrakten Auftrags überprüfen zu lassen, ob der Auftrag im Kopf der Warteschlange *RQ* (also der nächste vom Adapter zu bearbeitende Auftrag) dringend geworden ist und, wenn ja, den Adapter gezielt zu aktivieren.

Wird Δt in der Definition des dringenden Auftrags genau auf die Bearbeitungszeit T_{ADAPT} des Auftrags in dem Adapter gesetzt, steht der Auftrag im Adapter um T_{ADAPT} sek. früher zur Verfügung und kann gerade rechtzeitig an der *IF* übergeben werden. Das Δt bzw. die Bearbeitungszeit T_{ADAPT} kann z.B. durch die durchschnittliche Auftragsbearbeitungszeit $T_{MEAN,\ ADAPT}$ in dem Adapter angenähert werden (ermittelt entweder als Durchschnitt über alle Werte oder ggf. mithilfe geeigneter Fenstertechniken, siehe z.B. [WHW05]).

Nach der Übergabe des Auftrags an *IF* sollte der Adapter die Kontrolle an den Generator zurückgeben, es sei denn, der nächste zu bearbeitende Auftrag ist inzwischen dringend geworden oder der Generator befindet sich im blockierten Makrozustand φ_b des BVA. Der Generator wäre nur dann aktiv, wenn:

1) keine dringenden Aufträge in der *RQ* sind und
2) die *RQ* nicht voll ist und
3) der aktuelle Zustand nicht in dem blockierten Makrozustand φ_b des BVA liegt.

Der Adapter ist aktiv, wenn:

1) dringende Aufträge in der *RQ* vorliegen oder
2) die *RQ* voll ist oder
3) der Generator sich in dem blockierten Makrozustand φ_b des BVA befindet.

Ein möglicher Wechsel der Aktivphasen des Generators und des Adapters im Lastgenerierungsprozess ist in der Abb. 4.3 exemplarisch dargestellt. Die horizontalen Linien stellen jeweils die Zeitachsen für die physikalische Zeit in dem Generator und dem Adapter dar, die aktiven Phasen des Generators sind blau (grau) und die des Adapters sind grün

(dunkel schraffiert) markiert. Die Phasen, in denen der Generator konzeptionell aktiv (der aktive Makrozustand φ_a des BVA), programmtechnisch aber passiv ist, sind hellblau (hell schraffiert) dargestellt.

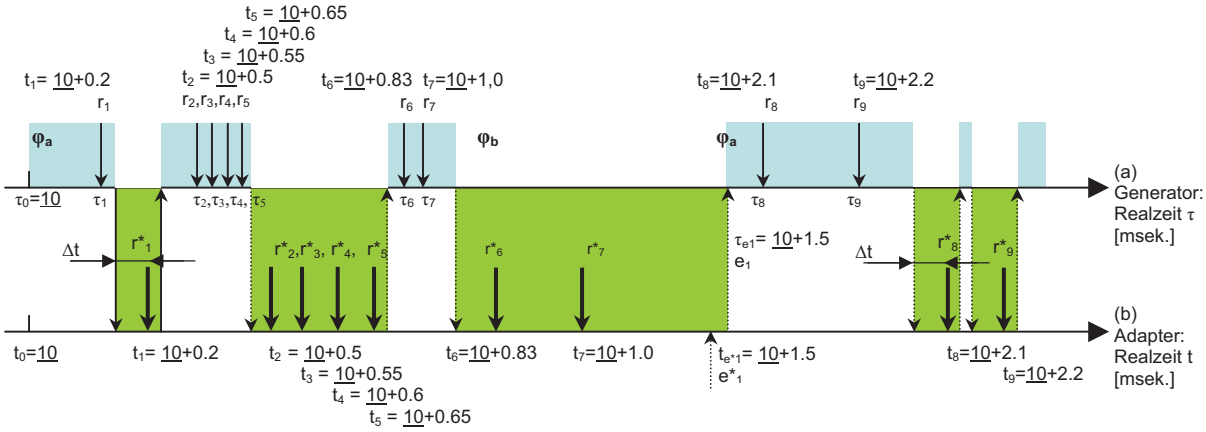


Abbildung 4.3: Sichere Kontrollübergabe zwischen dem Generator und dem Adapter

Der Adapter wird vor dem Generator erzeugt und in einen passiven Zustand versetzt, in dem er auf die Aufträge in der *RQ* wartet. Danach wird der Generator erzeugt. Nach dem Start befindet er sich zunächst in dem Initialisierungszustand φ_i des BVA. Der Generator und der Adapter verwenden gemeinsam eine Systemuhr um den physikalischen Startzeitpunkt τ_0 (bzw. $t_0 = \tau_0$ im Adapter) des Lastgenerierungsprozesses und die aktuelle physikalische Systemzeit t_{NOW} zu bestimmen.

Zum Zeitpunkt τ_0 beginnt der Generator seine erste Aktivphase, er verlässt den Initialisierungszustand φ_i und geht in den aktiven Makrozustand φ_a des BVA. In den D-Zuständen des BVA wird die logische Uhr des Generators gemäß der spezifizierten Verzögerungs- bzw. Zwischenankunftszeit fortgeschaltet. In den R-Zuständen generiert er abstrakte Aufträge und verwendet den Wert seiner logischen Uhr plus τ_0 als Übergabezeitpunkt des nächsten Auftrags. Wenn der nächste zu bearbeitende Auftrag (t_1, r_1) im Kopf der *RQ* dringend geworden ist ($[t_1 - t_{NOW} \leq \Delta t]$ ist erfüllt), beendet der Generator seine erste Aktivphase und aktiviert den Adapter.

In seiner ersten Aktivphase bereitet der Adapter den realen Auftrag (t_1, r^*_1) vor, und übergibt ihn sofort an *IF*. Danach stellt er fest, dass keine weiteren dringenden Aufträge in der *RQ* vorliegen und übergibt die Kontrolle zurück an den Generator.

In der Aktivphase 2 befindet sich der Generator wieterhin in dem aktiven Makrozustand φ_a des BVA. Nun generiert er einen Burst von N Aufträgen (t_i, r_i), i = 1, …, N, wobei N die Kapazität der Warteschlange *RQ* ist. Der Generator stellt fest, dass die *RQ* voll ist, und übergibt die Kontrolle an den Adapter. Der Adapter bleibt solange aktiv, bis mindestens ein Auftrag aus der *RQ* bearbeitet ist. Danach bearbeitet er entweder weitere Aufträge aus der *RQ*, falls sie dringend geworden sind, oder er gibt die Kontrolle an den Generator zurück.

In der Aktivphase 3 generiert der Generator die Aufträge (t_6, r_6) und (t_7, r_7) und geht anschließend in den blockierten Makrozustand φ_b des BVA. Die Kontrolle geht an den Adapter über. Soweit dringende Aufträge vorliegen, werden sie vom Adapter bearbeitet. Ansonsten ist der Adapter solange aktiv, bis die erwartete Systemreaktion (t_{e^*1}, e^*_1) vorliegt (bzw. bis die maximale Blockierungszeit verstrichen ist). Der Adapter bereitet die entsprechende abstrakte Reaktionsnachricht (τ_{e1}, e_1) vor, stellt sie in die Warteschlange *EQ* ein und aktiviert den Generator.

In der Aktivphase 4 verarbeitet der Generator die Nachricht aus der *EQ* (indem er den Nachfolgerzustand in dem BVA entsprechend auswählt). Danach ist er wieder in dem aktiven Makrozustand φ_a des BVA. Er produziert die letzten Aufträge (t_8, r_8) und (t_9, r_9) und geht in den Terminierungszustand φ_t des BVA. Der Generator ist damit beendet und die Kontrolle wird an den Adapter übergeben, der noch weiter ausstehende Aufträge aus der *RQ* bearbeiten soll. Nach der Bearbeitung des letzten Auftrags (t_9, r_9) terminiert der Adapter.

Es ist zu beachten, dass die Prüfung der dringenden Aufträge in der *RQ* nicht nur von dem Generator, sondern auch von dem Adapter in seinen aktiven Phasen durchgeführt wird. Der Adapter überprüft nach der Bearbeitung jedes Auftrags ebenfalls, ob weitere Aufträge in der *RQ* inzwischen dringend geworden sind und behält die Kontrolle bei, solange solche dringenden Aufträge in der *RQ* noch vorliegen.

5. Aspekte der Realisierung des Lastgenerators

Der in Abschnitt 4 skizzierte Entwurf wurde in einer ersten echtzeitfähigen Version des Lastgenerators UniLoG in Verbindung mit einem Adapter für die Schnittstellen UDP und TCP mithilfe von Threads unter Microsoft Windows XP (SP2) realisiert. Für die Generierung von UDP- und TCP-Auf-

trägen im Adapter wurden Windows Sockets (die dem Paradigma der BSD-Sockets im Wesentlichen entsprechen) in der Version 2.2 verwendet.

Die bestehende konzeptionelle Aufgabenteilung zwischen dem Generator und dem Adapter wird auf den Generator-Thread (zuständig für die Auswertung des PBVA in Echtzeit) und den Adapter-Thread (zuständig für die Generierung von UDP- und TCP-Aufträgen und Modellierung von Systemreaktionen) übertragen, die auf einem Prozessor quasi-parallel ausgeführt werden. Durch die Auslagerung der netzabhängigen Aktivitäten in einen separaten Adapter-Thread wird die Berücksichtigung von Reaktionen später möglich, die in einem externen Netzmodell generiert worden sind.

Threads sind ein Scheduling-Konzept und bieten gegenüber Prozessen den Vorteil einer weniger aufwendigen Kontrollübergabe. Diesen Umstand nutzen wir aus, um die Kontrolle schnell an den Adapter-Thread zu übergeben, falls der nächste zu bearbeitende Auftrag dringend geworden ist (siehe dazu Abschnitt 5.3). Da jeder Thread auf alle Objekte zugreifen darf, die zu seinem Prozess gehören, wurden die Warteschlangen *RQ* und *EQ* als globale Objekte in dem für den Generator- und den Adapter-Thread gemeinsamen Adressraum realisiert.

Zur Synchronisation von Threads bietet Windows XP eine ganze Reihe von Primitiven an (wie z.B. kritische Bereiche, Events, Semaphoren, Mutexe, etc.). Wir verwendeten Mutexe für den wechselseitigen Ausschluss beim Zugriff auf die Warteschlangen und Semaphoren für die explizite Synchronisation (Kontrollübergabe mit Berücksichtigung der Dringlichkeit der Aufträge) zwischen dem Generator- und dem Adapter-Thread.

Die Benutzung der Synchronisationsprimitiven Mutex und Semaphor folgt unter Windows einem bestimmten Muster. Um z.B. einen Mutex mxRQ für die *RQ* anzufordern, ruft der Generator-Thread die Win32 API Wartefunktion WaitForSingleObject(mxRQ) auf. Wenn der Adapter-Thread gerade in Besitz von Mutex ist (mxRQ ist damit 0 = "nicht signalisiert"), blockiert der Generator-Thread in der Wartefunktion, bis der Adapter-Thread den Mutex mit ReleaseMutex() freigibt (auf 1 = "signalisiert" setzt). Ist der Mutex "signalisiert" wird er atomar auf 0 = "nicht signalisiert" zurückgesetzt und der Generator-Thread verlässt die Wartefunktion und tritt in seinen kritischen Bereich ein. Ein auf einen Mutex bzw. einen Semaphor wartender Thread verbraucht dabei keine Prozessorzeit [MSDN].

Für die möglichst exakte Bestimmung der Übergabezeitpunkte der UDP- und TCP-Aufträge werden Uhrprimitiven mit der Genauigkeit im Bereich weniger µs benötigt. Windows stellt in Win32 API mit QueryPerformanceFrequency() und QueryPerformanceCounter() eine solche Uhr zur Verfügung. Bei der Entwicklung des Werkzeugs UniLoG hat sich der Einsatz von kurzen aktiven Warteschleifen mit Aufruf von QueryPerformanceCounter() als sehr effektive Methode zur Bestimmung der Auftragsübergabezeitpunkte erwiesen.

In der neuen echtzeitfähigen Version des Lastgenerators erhöhen wir zusätzlich die Priorität des Adapter-Threads (um den Einfluss der betriebssystembedingten Faktoren zu minimieren) und nehmen gezielt Einfluss auf die Dauer der aktiven Warteschleifen durch eine explizite Synchronisationsmethode. Für die Generierung von Aufträgen erforderliche Modellkomponenten und Parameter werden möglichst in der Initialisierungsphase geladen, um ihr Nachladen während der Auftragsgenerierung auszuschließen.

5.1 Der Generator-Thread

Der Generator-Thread ist zuständig für die Ausführung des PBVA in Echtzeit. Die Hauptschleife des Generator-Threads enthält die Operationen zur Ausführung von Aktionen in den Zuständen des PBVA und wird zusätzlich mit der Funktion zur sicheren Kontrollübergabe an den Adapter für die Echtzeitfähigkeit ergänzt (siehe Abschnitt 5.3).

Der Generator-Thread verlässt den Initialisierungszustand φ_i des PBVA zum physikalischen Zeitpunkt τ_0 und wird fortgesetzt, solange der Terminierungszustand φ_t nicht erreicht und die maximale Lastgenerierungsdauer nicht überschritten wurde. Die logische Zeit wird in den D-Zuständen gemäß Modellparametern fortgeschaltet. Der Übergabezeitpunkt des nächsten abstrakten Auftrags in den R-Zuständen wird auf die aktuelle physikalische Zeit gesetzt, die sich als (τ_0 + akt. logische Zeit) ergibt.

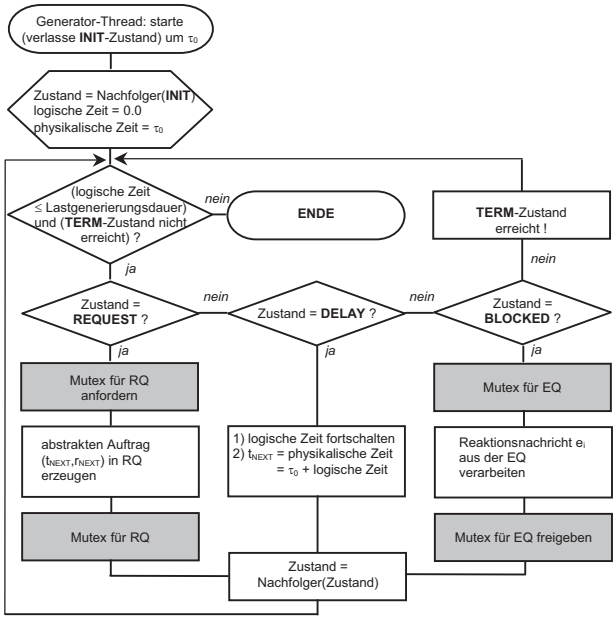


Abbildung 5.1: Hauptschleife des Generator-Threads

Die Zugriffe auf die Warteschlangen *RQ* und *EQ* erfolgen im wechselseitigen Ausschluss mit dem Adapter und sind jeweils mit einem Mutex für *RQ* und einem Mutex für *EQ* geschützt. Wenn der Generator in Besitz des entsprechenden Mutex ist, überprüft er zunächst, ob *RQ* voll bzw. *EQ* leer ist, um einen möglichen illegalen Zustand zu vermeiden.

5.2 Der Adapter-Thread

Der Adapter-Thread wird zum Zeitpunkt $t < t_0$ gestartet ($t_0 = \tau_0$ ist der Startzeitpunkt des Lastgenerierungsprozesses im Generator) und fortgesetzt, solange der Lastgenerierungshorizont (t_0 + Lastgenerierungsdauer) nicht erreicht ist. Soweit keine Nachricht vom Generator-Thread über die Blockierung in dem Makrozustand φ_b vorliegt, bearbeitet der Adapter-Thread die abstrakten Aufträge aus der *RQ*, anderenfalls führt er die Operationen zur Modellierung der Systemreaktion aus.

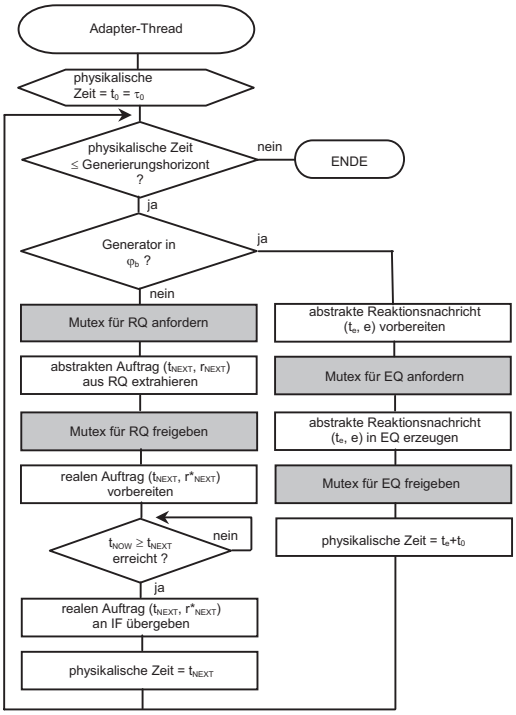


Abbildung 5.2: Hauptschleife des Adapter-Threads

Im Folgenden gehen wir kurz auf den Adapter für UDP ein. Die Pflichtattribute eines UDP-Auftrags sind die IP- und die Portnummer des Senders und des Empfängers sowie die zu übermittelnden Nutzdaten. Falls in den abstrakten Aufträgen (t_{NEXT}, r_{NEXT}) in der *RQ* fehlend, werden diese Attribute in den realen UDP-Aufträgen (t_{NEXT}, r^*_{NEXT}) beim Aufruf von sendto() mit spezifizierten Standardwerten aus dem PBVA vorbelegt (z.B. wird ein Bitmuster gewählt, falls nur die Datenlänge als Attribut gegeben wurde).

Zur Bestimmung des Übergabezeitpunktes der UDP-Aufträge werden kurze aktive Warteschleifen (mit dem Aufruf der Win32 API-Funktion GetPerformanceCounter()) eingesetzt. Die Thread-Priorität des Adapter-Threads wird hier mit SetThreadPriority() um 2 Punkte über der normalen Priorität angehoben, um die Einflüsse anderer Threads im System zu minimieren. Andere ressourcen-sparende Uhrprimitiven (z.B. unterbrechbare Timer) sind an dieser Stelle leider nicht geeignet, da ihre Auflösung an die Interrupt-Periode des Schedulers gebunden und damit für unseren Fall viel zu grob ist (z.B. werden die Interrupts bei Windows XP SP2 ca. alle 10ms ausgelöst, s. [MSDN]).

Bei der Modellierung von Systemreaktion ist es dagegen zu erwarten, dass die systeminduzierten Blockierungszeiten des Benutzers im Bereich von ms liegen. Wenn die Blockierungszeit vorher bestimmt und größer als die Interrupt-Periode des Schedulers (10ms) ist, kann in dem Adapter-Thread ein Timer für das größte Vielfache von 10ms, das kleiner der Blockierungsdauer ist, gesetzt werden. Damit würde der Adapter-Thread auch in den Blockierungsphasen des Generators keine Prozessorzeit unnötig verbrauchen.

Die im nächsten Abschnitt beschriebene Synchronisationsfunktion erlaubt die aktiven Wartephasen im Adapter zu verkürzen.

5.3 Synchronisation zwischen dem Generator-Thread und dem Adapter-Thread

Zur Realisierung der Synchronisation zwischen dem Generator-Thread und dem Adapter-Thread werden zwei binäre Semaphoren generatorAktiv (initialisiert mit 1, "signalisiert") und adapterAktiv (initialisiert mit 0, "nicht signalisiert") verwendet. Der Adapter-Thread wird zuerst gestartet und bleibt in dem Aufruf von WaitForSingleObject() blockiert, bis der Generator-Thread den Wert des Semaphors adapterAktiv mit ReleaseSemaphore() auf 1 ("signalisiert") erhöht (siehe Abb. 5.3-5.4).

Da der Semaphor generatorAktiv mit 1 ("signalisiert") initialisiert wurde, kann der Generator-Thread seinen Aufruf von WaitForSingleObject() passieren, wobei der Wert von generatorAktiv im Aufruf WaitForSingleObject() atomar auf 0 ("nicht signalisiert") gesetzt wird. Der Generator kann nun die Operationen in dem aktuellen Zustand des PBVA ausführen (z.B. er generiert den nächsten Auftrag (t_{NEXT}, r_{NEXT})) und bestimmt den Nachfolgerzustand im PBVA (z.B. einen Zustand aus dem blockierten Makrozustand φ_b).

Der Generator-Thread überprüft anschließend, ob

1) *RQ* voll geworden ist oder
2) der neue Zustand im blockierten Makrozustand φ_b des PBVA liegt bzw.
3) der nächste vom Adapter zu bearbeitende Auftrag (t_{NEXT}, r_{NEXT}) dringend geworden ist (t_{NOW}-$t_{NEXT} < \Delta t$).

Falls keine dieser Bedingungen erfüllt ist, wird der Wert des Semaphors generatorAktiv wieder auf 1 ("signalisiert") erhöht, so dass der Generator-Thread den Aufruf von WaitForSingleObject() passieren kann und in die nächste Iteration seiner Hauptschleife gelangt.

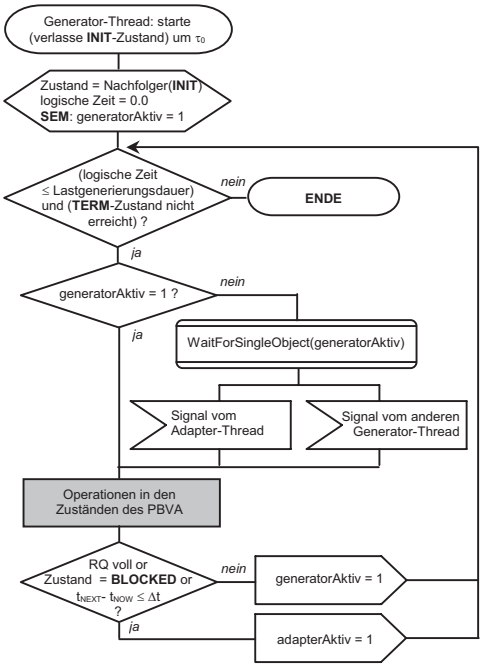


Abbildung 5.3: Kontrollübergabe in dem Generator-Thread

Anderenfalls wird der Semaphor adapterAktiv signalisiert und der Generator-Thread blockiert in seinem nächsten Aufruf von WaitForSingleObject(), da generatorAktiv 0 ("nicht signalisiert") ist. An dieser Stelle wird der Scheduler des Betriebssystems aufgerufen, der den Adapter-Thread aus dem blockierten Aufruf von WaitForSingleObject() löst.

Der Adapter-Thread gelangt in seine Hauptschleife und kann den nächsten Auftrag aus der *RQ* bearbeiten oder die Systemreaktion modellieren, wenn die Nachricht über die Blockierung des Generators in φ_b bereits vorliegt (siehe Abb. 5.4). Nach der Generierung des nächsten UDP- bzw. TCP-Auftrags überprüft der Adapter-Thread, ob die *RQ* leer geworden ist, und, wenn ja, aktiviert er den Generator-Thread, indem er den Semaphor generatorAktiv signalisiert (auf 1 erhöht).

Wenn die *RQ* nicht leer ist, überprüft der Adapter, ob der nächste zu bearbeitende abstrakte Auftrag (t_{NEXT}, r_{NEXT}) ebenfalls dringend geworden ist und behält bzw. gibt die Kontrolle an den Generator-Thread ab, indem er den Semaphor adapterAktiv bzw. generatorAktiv signalisiert. Nach der Erzeugung der abstrakten Reaktionsnachricht und Modellierung der Blockierungsdauer signalisiert der Adapter-Thread zwingend den Semaphor generatorAktiv.

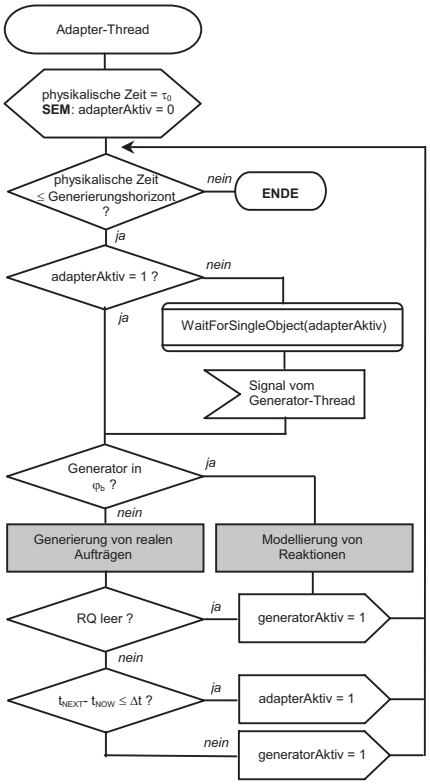


Abbildung 5.4: Kontrollübergabe in dem Adapter-Thread

Bei allen möglichen Varianten der PBVA sollte die Termination der beiden Threads sichergestellt werden. Der Adapter könnte z.B. eine Reaktionsnachricht für den Zeitpunkt generieren, der bereits hinter dem Lastgenerierungshorizont liegt, und würde versuchen, den bereits terminierten Generator-Thread zu aktivieren. In solchen Fällen wird eine Sonderbehandlung vorgenommen, die in den Abbildungen 5.3-5.4 zur Vereinfachung nicht eingezeichnet wurde.

Mit dem Parameter Δt kann somit gesteuert werden, um wie viele µs früher der Generator-Thread die Kontrolle an den Adapter-Thread übergibt. Damit lässt sich die auftragslängenabhängige Bearbeitungszeit in dem Adapter-Thread T_{ADAPT} berücksichtigen. Bei konstanter Auftragslänge ist die Bearbeitungszeit in dem Adapter ebenfalls konstant, so sollte $\Delta t = T_{ADAPT}$ gewählt werden. Bei variabler Auftragslänge könnte Δt durch die mittlere Bearbeitungszeit $T_{MEAN,\ ADAPT}$ angenähert werden (vgl. Abschn. 4.3).

6. Experimenteller Teil

Die Hauptkriterien für die Bewertung der Leistungsfähigkeit bzw. der Präzision des realisierten Lastgenerators sind die maximale erreichbare Da-

tenrate der generierten Verkehrsströme bzw. der Mittelwert und die Varianz der Verteilung von Differenzen zwischen den spezifizierten und den faktischen Übergabezeitpunkten der realen Aufträge. Diese Kriterien haben einen strengen Bezug zu der Schnittstelle *IF*, die von dem Experimentator zur Lastgenerierung gewählt wurde.

Um die ersten Aussagen zur Leistungsfähigkeit und zur Präzision des neuen Lastgenerators zunächst in Verbindung mit dem realisierten UDP-Adapter zu treffen, wurden Experimente zur Generierung von künstlichen UDP-Verkehrsströmen mit CBR (constant bit rate) in einem 100 MBit/s Fast Ethernet LAN mit einem Hub TP800 der Firma 3Com bei einer MTU-Größe von 1500 Byte durchgeführt. Der Lastgenerator wurde auf einem handelsüblichen PC mit Intel Pentium 4 CPU 2.39 GHz, 1,00 GB RAM, Intel PRO/100 VM Network Connection Adapter, Microsoft Windows XP Professional, Version 2002, Service Pack 2 ausgeführt. Als Lastsenke für den generierten UDP-Strom kam ein PC mit einer identischen Systemkonfiguration zum Einsatz. Ein dritter Windows-PC wurde für die Aufzeichnung und die Analyse des Verkehrs im LAN mit dem Netzwerk-Analysator Ethereal (Vers. 0.99) verwendet.

Die mittlere Datenrate eines UDP-Stroms kann zum einen durch die Veränderung der Auftragslänge L und zum anderen durch die Veränderung der Zwischenankunftszeit (ZAZ) der Aufträge beeinflusst werden. In der ersten Experimentserie wurde die Auftragslänge absichtlich groß L_1 = 10000[Byte] gewählt und die ZAZ sehr nah an dem rechnerisch möglichen Minimum von 1[ms] bis 800[µs] systematisch variiert[3].

ZAZ [µs]	1000	900	890	870	850	(840)	(830)
# Aufträge	29998	33332	33706	34481	35292	35713	36144
ΔT_{mean} (µs)	8	11	14	25	45	1815	192338
Var(ΔT)	0.002	0.0035	0.0038	0.008	0.014	0.821	12776
ΔT >= 1ms	1	3	0	21	35	27762	36081
ΔT ∈ [800, 1000) µs	5	16	7	27	55	3126	25
ΔT ∈ [600, 800) µs	17	37	35	76	211	2941	7
ΔT ∈ [400, 600) µs	29	71	95	251	586	1231	4
ΔT ∈ [200, 400) µs	434	689	933	1401	2296	526	23
ΔT ∈ [100, 200) µs	435	561	757	1199	2141	127	4
ΔT ∈ [90, 100) µs	66	71	106	133	221	0	0
ΔT ∈ [80, 90) µs	64	76	75	124	206	0	0
ΔT ∈ [70, 80) µs	128	155	153	175	260	0	0
ΔT ∈ [60, 70) µs	197	186	222	214	299	0	0
ΔT ∈ [50, 60) µs	245	167	275	250	327	0	0
ΔT ∈ [40, 50) µs	412	448	445	476	471	0	0
ΔT ∈ [30, 40) µs	165	210	214	276	390	0	0
ΔT ∈ [20, 30) µs	284	268	258	304	401	0	0
ΔT ∈ [10, 20) µs	407	338	319	329	529	0	0
ΔT ∈ [5, 10) µs	552	345	366	458	502	0	0
ΔT < 5µs	26557	29691	29446	28767	26362	0	0
% (ΔT < 10 µs)	90.37	90.11	88.45	84.76	76.12	0	0

Tabelle 6.1: Experimentserie 1, L_1 = 10000[Byte], ZAZ = 1000 ÷ 830[µs], Lastgenerierungsdauer 30[sek.].

Die Lastgenerierungsdauer wurde konstant bei 30[sek.] gehalten. In jedem Experiment wurde für jeden Auftrag die Abweichung $\Delta T = t_{IST} - t_{SOLL}$ zwischen dem spezifizierten Übergabezeitpunkt t_{SOLL} und dem gemessenen faktischen Übergabezeitpunkt t_{IST} berechnet. Die Anzahl der generierten UDP-Aufträge (#Aufträge), die mittlere Differenz der faktischen und der spezifizierten Übergabezeitpunkte der Aufträge (ΔT_{mean}), ihre Varianz Var(ΔT) sowie die Häufigkeitsverteilung von ΔT (die sog. "Ausreißer") sind in der Tabelle 6.1 für verschiedene Werte der ZAZ zusammengefasst.

Es ist zu bemerken, dass bei der ZAZ von 1[ms] und 900[µs] etwas mehr als 90% der Differenzen ΔT kleiner 10[µs] sind (vgl. die letzte Zeile in der Tab. 6.1). Bei der ZAZ von 890, 870 und 850 [ms] sind es jeweils 88%, 84% und 76%. Die relativ hohe Anzahl und eine breite Streuung von Ausreißern dürfte in dem Fall L = 10000[Byte] auf den Einfluss der Fragmentierung und der damit verbundenen Verarbeitungsprozesse in der IP-Schicht zurückzuführen sein.

ZAZ [µs]	500	300	150	140	(127)	(126)	(125)	(120)
#Aufträge	59998	99998	199997	214282	236217	238092	239997	249997
ΔT_{mean} (µs)	3	6	10	13	807	3120	27164	649369
Var(ΔT)	0.0005	0.0012	0.0024	0.0042	8.04	39.93	922.62	136787
ΔT >= 1ms	0	0	23	153	22546	60430	114000	249817
ΔT ∈ [800, 1000) µs	7	8	63	69	1462	1336	1628	15
ΔT ∈ [600, 800) µs	16	25	139	217	2790	2866	3815	52
ΔT ∈ [400, 600) µs	15	22	237	486	6469	7356	10404	51
ΔT ∈ [200, 400) µs	132	1014	3040	4732	13961	15192	18095	50
ΔT ∈ [100, 200) µs	261	1170	3418	4652	7923	8372	9625	8
ΔT ∈ [90, 100) µs	23	84	249	488	793	871	981	4
ΔT ∈ [80, 90) µs	8	159	586	484	811	836	961	0
ΔT ∈ [70, 80) µs	11	182	370	541	828	842	948	0
ΔT ∈ [60, 70) µs	23	352	410	593	820	854	967	0
ΔT ∈ [50, 60) µs	48	333	499	518	841	894	992	0
ΔT ∈ [40, 50) µs	69	380	459	519	912	945	1016	0
ΔT ∈ [30, 40) µs	105	414	589	568	888	948	1066	0
ΔT ∈ [20, 30) µs	262	403	519	638	1030	1040	1171	0
ΔT ∈ [10, 20) µs	1314	929	966	1090	1471	1486	1542	0
ΔT ∈ [5, 10) µs	3970	2895	3947	3845	4004	3717	3669	0
ΔT < 5 µs	53734	91628	184483	194689	168668	130107	69117	0
% (ΔT < 10 µs)	96.18	94.52	94.22	92.65	73.1	56.21	30.32	0

Tabelle 6.2: Experimentserie 2, L = 1472[Byte], ZAZ = 500 ÷ 120[µs], Lastgenerierungsdauer 30[sek.].

Aus diesen Gründen wurde die zweite Experimentserie mit der größten möglichen Auftragslänge durchgeführt, bei der noch keine Fragmentierung der Pakete in der IP-Schicht stattfindet. Bei der gegebenen MTU-Größe von 1500 Byte ergibt sich die entsprechende maximale Auftragslänge L_2 von 1472[Byte]: 1500[Byte] {MTU} - 20[Byte] {IP-Header} - 8[Byte] {UDP-Header}. Die ZAZ der Aufträge wurde von 500[µs] bis 120[µs] variiert[4]. Die Experimentergebnisse sind in der Tabelle 6.2 zusammengefasst.

[3] Für die Übergabe von L_1 = 10000[Byte] Nutzdaten an der UDP-Schnittstelle im gegebenen 100 Mbit/s Fast Ethernet LAN werden gerade mindestens ca. 800[µs] benötigt: ((10000[Byte]{Nutzdaten} +8[Byte]{UDP-Header} + 20[Byte]{IP-Header}+14[Byte]{MAC-Header}) * 8[Bit/Byte] / 100[MBit/s] = 803.36[µs]).

[4] Für die Übergabe von L_2 = 1472[Byte] Nutzdaten an der UDP-Schnittstelle im gegebenen 100 Mbit/s Fast Ethernet LAN werden mindestens 120[µs] benötigt: ((1472[Byte]{Nutzdaten} + 8[Byte]{UDP-Header} + 20[Byte]{IP-Header} + 14[Byte]{MAC-Header}) * 8[Bit/Byte] / 100[MBit/s] = 121.12[µs])

Obwohl im Vergleich zu der Experimentserie 1 fast 10 Mal so viele Aufträge erzeugt werden, liegen bei den ZAZ im Bereich von 500 bis ca. 150 [µs] mehr als 94% der Werte von ΔT unter 10[µs] (siehe Tabelle 6.2). Die im Vergleich zur Experimentserie 1 kleinere relative Anzahl von Ausreißern lässt sich vermutlich durch die bei L_2=1472[Byte] nicht mehr stattfindende Fragmentierung erklären.

Die Abhängigkeit der Größe ΔT_{mean} von der spezifizierten Zwischenankunftszeit ZAZ und der Länge L der Aufträge wurde in der Abb. 6.3 dargestellt. Bei festem L beobachtet man einen rapiden Anstieg von ΔT_{mean}, je näher die ZAZ am rechnerischen Übergabezeitlimit der Aufträge an der UDP-Schnittstelle spezifiziert wird (803[µs] bei L_1 = 10000 [Byte] bzw. 121[µs] bei L_2 = 1472[Byte]).

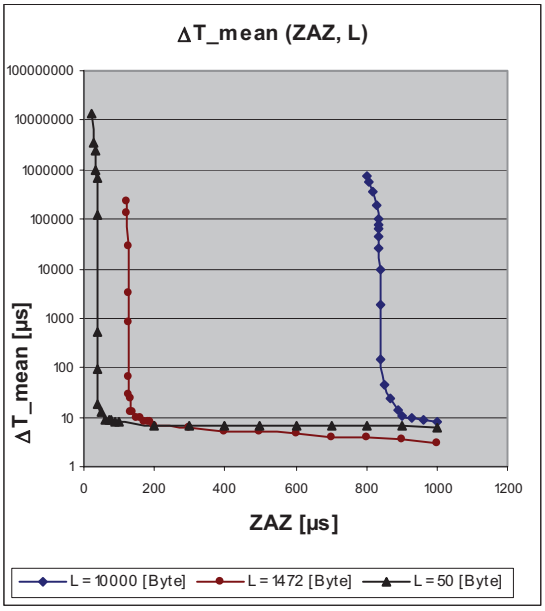


Abbildung 6.3: Mittlere Differenz ΔT_{MEAN} der spezifizierten und der faktischen Übergabezeitpunkte der UDP-Aufträge bei L_1 = 10000, L_2 = 1472, L_3 = 50 [Byte].

Somit lassen sich mit UniLoG UDP-Verkehrsströme mit einer konstanten mittleren Datenrate bis zu 80[Mbit/s] (und ggf. noch höher) auf der UDP-Schicht erzeugen. Die Systemauslastung auf der Ethernet-Ebene ist wegen der hinzugefügten UDP-, IP- und MAC-Header noch etwas höher. Die in den Experimentserien induzierte Netzauslastung wurde durch die Analysen des protokollierten LAN-Verkehrs in Ethereal sowie durch die Leuchtanzeigen verschiedener Auslastungsniveaus auf dem eingesetzten Hub bestätigt. Bei der Auftragslänge L_2 = 1472[Byte] und der ZAZ von 140[µs] befand sich das Netz bereits im Überlastbereich (86% Systemauslastung).

7. Zusammenfassung und Ausblick

In dem vorliegenden Beitrag wurde ausgehend von einer bestehenden Architektur eines verallgemeinerten Lastgenerators ein Entwurf für einen neuen echtzeitfähigen Lastgenerator UniLoG vorgeschlagen. In dem neuen Entwurf wurde die Möglichkeit der netzbedingten Blockierungszustände des Benutzers berücksichtigt. Die für eine Echtzeitumgebung kritischen Punkte wurden diskutiert und die entsprechenden Möglichkeiten zu ihrer Lösung erläutert.

Der vorgeschlagene Entwurf wurde in einer Multithread-Umgebung unter Microsoft Windows mit jeweils einem Adapter für die UDP- und die TCP-Schnittstellen realisiert. Für die möglichst exakte Einhaltung der spezifizierten Übergabezeitpunkte der UDP- und der TCP-Aufträge wurde eine spezielle Synchronisationsfunktion zur sicheren Kontrollübergabe zwischen dem Generator- und dem Adapter-Thread implementiert.

Die erste Version des neuen Lastgenerators UniLoG lieferte in Verbindung mit dem neuen realisierten UDP-Adapter bereits sehr ermutigende Ergebnisse bzgl. der erreichbaren Datenrate (bis zu 80[MBit/s]) und der Auslieferungsgenauigkeit des zu generierenden UDP-Auftragsstroms (95% der Differenzwerte ΔT zwischen den Soll- und Ist-Übergabezeitpunkten waren kleiner als 10[µs]).

An der Arbeitsgruppe TKRN der Universität Hamburg sind derzeit zusätzliche Erweiterungen der bisherigen Realisierung von UniLoG geplant:

- Erweiterung des Entwurfs auf den Fall mehrerer lastgenerierender Benutzer, Superposition mehrerer Verkehrsströme in dem UDP- und TCP-Adapter,
- Implementierung weiterer Adapter für die IP- und HTTP-Schnittstelle,
- Implementierung weiterer Transformationsarten wie z.B. Leaky-Bucket oder Token-Bucket in Echtzeit.

Mögliche Richtungen für zukünftige Forschungsarbeiten im Bereich der Lastgenerierung sehen wir überdies in:

- einer Integration von UniLoG in einen Netzemulator (z.B. *NetEmu*) zur Berücksichtigung von extern modellierten Systemreaktionen (in einem Netzemulator) in den Blockierungszuständen des Benutzers,
- der Einbindung von UniLoG in eine Umgebung zur verteilten Lastgenerierung,
- einer Vertiefung des bestehenden Konzepts eines Multi-Level Load Generators (*MLLG*) [CWZ03] durch die Realisierung von verschiedenen echtzeitfähigen Transformatoren.

Es ist zu hoffen und zu erwarten, dass das Werkzeug UniLoG nach der Realisierung der geplanten Erweiterungen zu einem praxisrelevanten Lastgenerierungstool mit einem breiten Anwendungsspektrum werden wird.

Danksagung

Ein großer Dank wird an dieser Stelle an Herrn Prof. Dr. Bernd E. Wolfinger sowie Herrn Stephan Heckmüller für die bei der Erstellung dieses Papiers geleistete Unterstützung ausgesprochen.

Literaturverzeichnis

[BaC98] Barford, P.; Crovella, M.: Generating representative Web workloads for network and server performance evaluation, Proceedings of the 1998 ACM SIGMETRICS joint international conference on Measurement and modeling of computer systems, p.151-160, June 22-26, 1998, Madison, Wisconsin, United States.

[Bai99] Bai, G.: Load Measurements and Load Modeling for Distributed Multimedia Applications in High-Speed Networks, Uni Press Hochschulschriften Bd. 107, auch: Dissertation, Fachbereich Informatik, Univ. Hamburg, 1999.

[BGP05] Bonelli, N.; Giordano, S.; Procissi, G. et al.: BRUTE: a High Performance and Extensible Traffic Generator. Proc. of SPECTS 2005, Simulation Series, Vol. 37, No. 3, 2005, 839--845.

[BuW02] Mudashiru, B.; Williamson, C.: ProWGen: a synthetic workload generation tool for simulation evaluation of web proxy caches, Computer Networks: The International Journal of Computer and Telecommunications Networking, v.38 n.6, p.779-794, 22 April 2002.

[CoK05] Cong, J.; Kolesnikov, A.: A Methodology for Load Generation Based on a Formal Load Specification Technique, in: [WoH05], S. 71-82.

[Con06] Cong, J.: Load Specification and Load Generation for Multimedia Traffic Load in Computer Networks, Dissertation, Department Informatik, Universität Hamburg, 2006, erschienen in: Wolfinger B.E. (Hrsg.), Berichte aus dem Forschungsschwerpunkt Telekommunikation und Rechnernetze, Band 5, Shaker-Verlag, Aachen, 2006.

[CoW06] Cong, J.; Wolfinger, B.E.: A Unified Load Generator Based on Formal Load Specification and Load Transformation, Value Tools 2006, First Intern. IEEE Conf. on Performance Evaluation Methodologies and Tools, Pisa, October 2006.

[CWZ03] Cong J.; Wolfinger B.E.; Zaddach M.: Design and Application of Multi-Layered Load Generators, 2nd IASTED Internat. Conf. on Communications, Internet and Information Technology (CIIT 2003), Scottsdale, Arizona/USA, Nov. 2003.

[FHW04] Fiolka, K.; Heidtmann K.; Wolfinger B.: Experimentelles und exploratives Lernen mit selbstentwickelten eLearning Werkzeugen im Bereich der Telematik, 2. Deutsche Fachtagung der GI (DELFI 2004), Paderborn, Sept. 2004.

[Hee00] Heegaard, P. E.: Gensyn -- a Generator of Synthetic Internet Traffic Used in QoS Experiments. 15th Nordic Teletraffic Seminar, August, 2000 -- Lund, Sweden, 22--24.

[Kam04] Kamel, T.: Gewinnung und Einsatz lastabhängiger Traces zur Emulation von Kommunikationsnetzen, Diplomarbeit, Department Informatik, Universität Hamburg, 2004.

[MoG03] Moatemri, F.; Grégoire, J-Ch.: MLTG Traffic Re/Generator. Proc. of SPECTS 2003, Simulation Series, Vol. 35, No. 4, 2003.

[MSCE] Microsoft Developer Network, Online-Dokumentation, Abschnitt zu Embedded Operating System Development, http://msdn2.microsoft.com/en-us/library/ms905511.aspx, Online Resource, zuletzt abgerufen 10.05.07.

[MSDN] Microsoft Developer Network, Online-Dokumentation, Abschnitt zur Synchronisation, http://msdn2.microsoft.com/en-us/library/ms686353.aspx, Online Resource, zuletzt abgerufen 10.05.07.

[RRB07] Rolland, C.; Ridoux, J.; Baynat, B.: LiTGen, a lightweight traffic generator: application to P2P and mail wireless traffic, Proceedings PAM 2007, Louvain-la-neuve, Belgien, April 2007.

[RTL] Hard-Real-Time Deterministic Solutions for Linux & BSD, http://www.fsmlabs.com, Online-Resource, zuletzt abgerufen 12.05.07.

[SBW05] Scherpe, C.; Brehmer I.; Wolfinger B.E.: IP-basierte Emulation von gekoppelten Rechnernetzen, in: [WoH05], S. 61- 70.

[Sch06] Scherpe, C.: Emulation gekoppelter Rechnernetze mit lastabhängigem Verzögerungs- und Verlust-verhalten, Shaker-Verlag, auch: Dissertation, Fachbereich Informatik, Univ. Hamburg, 2006.

[SoB04] Sommers, J.; Barford, P.: Self-configuring network traffic generation, Proceedings of the 4th ACM SIGCOMM conference on Internet measurement, October 25-27, 2004, Taormina, Sicily, Italy.

[Ste99] Stevens, W.R.: UNIX network programming, Vol. 2, Interprocess communications, Upper Saddle River, NJ, Prentice Hall, 1999.

[Tan03] Tanenbaum, A.: Moderne Betriebssysteme, 2. überarb. Aufl., übers. von Uwe Baumgarten, Pearson Education, München, 2003.

[TFCV03] Tang, W.; Fu, Y.; Cherkasova, L.; Vahdat, A.: MediSyn: a synthetic streaming media service workload generator, Proceedings of the 13th international workshop on Network and operating systems support for digital audio and video, June 01-03, 2003, Monterey, CA, USA.

[WHW05] Wolf J.; Heckmüller S.; Wolfinger B.E.: Dynamic Resource Reservation and QoS Management in IEEE 802.11e Networks, International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS), July 24-28, Cherry Hill, New Jersey, USA, 2005 pp. 149-161.

[WoH05] Wolfinger, B.; Heidtmann, K.: Leistungs-, Zuverlässigkeits- und Verlässlichkeitsbewertung von Kommunikationsnetzen und verteilten Systemen, 3. GI/ITG-Workshop MMBnet 2005, Bericht 263, Universität Hamburg, Fachbereich Informatik, 2005.

[Wol99] Wolfinger, B.: Characterization of Mixed Traffic Load in Service-Integrated Networks, Systems Science Journal, Vol. 25, No. 2 (1999), S. 65-86

Quality of Service (QoS) bei IPTV

von

Tadeus Uhl
Fachhochschule Flensburg
tadeus.uhl@fh-flensburg.de

Lars Diederichsen
ITD Holdorf
lars.diederichsen@fh-flensburg.de

Keywords: communication networks, communication services, communication protocols, network architecture, multimedia applications, IPTV, QoS, QoE, PEVQ, MPEG-4/AVC

Kurzfassung

Der Kern dieser Arbeit liegt in der Darstellung der theoretischen und praktischen Aspekte bei dem neuen Dienst IPTV. Der erste Teil dieser Arbeit hat einführenden Charakter und enthält Informationen, die zum Verständnis der neuen Technik beitragen. Zuerst werden Architektur und Protokollmodell für den Service IPTV vorgestellt. Im weiteren werden zwei Gruppen von Messmethoden zur Beurteilung der QoS erörtert. Die erste Gruppe arbeitet mit den Netzwerkparametern (Netzverzögerung, Netzverzögerung Jitter, Paketverluste, Programmreferenzuhr Jitter, Synchronisationsverlust, Kontinuitätsfehler u.a.) und die zweite basiert auf den psycho-visuellen Modellen des menschlichen Sehvermögens (PEVQ: Perceptual Evaluation of Video Quality). Der zweite Teil dieser Arbeit hat praktischen Charakter. In ihm wird das neu entworfene und aufgebaute Messsystem „TraceView IPTV“ vorgestellt. Mit dem System ist es zur Zeit möglich, eine objektive Beurteilung der QoS bei IPTV durchzuführen. Dies wird in mehreren praktischen Beispielen verdeutlicht. Dabei wird die Leistungsfähigkeit einer IP-Transportplattform unter Berücksichtigung des Dienstes IPTV unter Beweis gestellt. Die aus dieser Analyse resultierenden Erkenntnisse sind für die Praxis von besonderer Bedeutung. Die Arbeit endet mit einer Zusammenfassung.

1 Einführung

Immer mehr Unternehmen bieten heutzutage Fernsehübertragungen und interaktive TV-Dienste wie VoD (Video on Demand) auf Basis der DSL (Digital Subscriber Line) -Technik an. An dieser Stelle ist anzumerken, dass es sich um einen schnellen DSL-Anschluss handelt, über den unter Verwendung von einfachen Telefondoppeladern Fernsehsignale übertragen werden. Eine weitere, noch bessere Alternative für Abwicklung von solchen Diensten über die letzte Meile bietet die PON (Passive Optical Network)-Technik an. Unter Verwendung dieser Technik ist es möglich, in diesem Bereich Übertragungsraten von einigen Gbit/s zu erreichen.

Eine wichtige Frage bei Fernsehübertragungen, wenn nicht sogar die wichtigste, ist die nach der Qualität. Während beim Surfen im Internet schon mal eine kleine Verzögerung akzeptiert wird, gelten für das Fernsehen über das IP-Netz höchste Ansprüche an Verfügbarkeit, Ton- und Bildqualität. Sie entscheiden über die Akzeptanz bei den Anwendern.

Durch entsprechende Tests können in einer IP-Plattform Probleme identifiziert und behoben werden. So können sich die Netzbetreiber ein klares Bild der QoE (Quality of Experience) ihrer Kunden vor der Bereitstellung und während des Betriebes machen. Ohne solche Informationen ist es sehr schwer, den Einfluss von IPTV auf den Datendurchsatz beim Internetsurfen oder Telefonieren und umgekehrt zu beurteilen. Untersuchungen haben ergeben, dass die Mehrheit der Fehler nicht im physikalischen Bereich auftaucht, sondern im Dienstsektor. Das zeigt, dass es bei IPTV erforderlich ist, die Dienste-Ebene genau so wie die physikalische Ebene zu betrachten. Es muss also nicht nur der Datentransfer untersucht werden, sondern vor allem auch, wie gut ein Dienst funktioniert. Hier ist vor allem eine effiziente Messtechnik zur Beurteilung der QoS (Quality of Service) und der QoE gefragt. Diesem Thema ist der Kern dieser Arbeit gewidmet.

2 Architektur für IPTV

Die Abb. 1 stellt eine typische IPTV-Architektur dar [1].

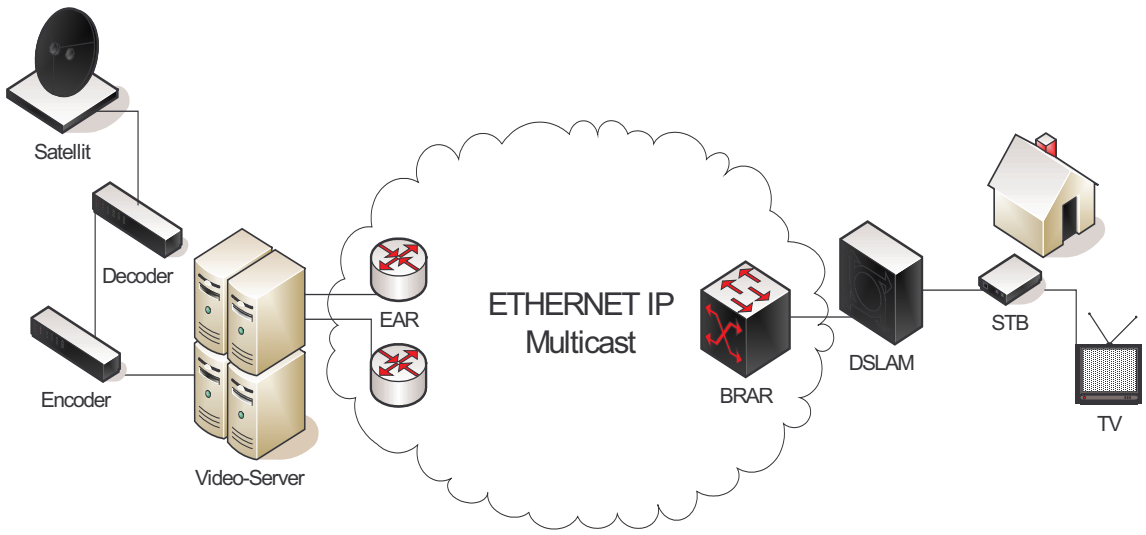


Abb. 1: Typische IPTV-Architektur

Der linke Teil der Abb. 1 stellt das so genannte Head End dar, das für die Aufbereitung und die Verteilung der Programme ins Netz verantwortlich ist. Die audiovisuellen Datenströme werden hier über breitbandige Anbindungen (ATM), über Kabel oder Satellit empfangen. Die empfangenen Signale werden demoduliert, descrambelt und ggf. decodiert und wieder encodiert. Danach werden die Datenströme an die Video-Server weitergeleitet, von wo aus sie an die Ethernet Access Router (EAR) gesendet werden. Durch das Multicast-Netz werden die Daten dann zu den Broadband Remote Access Routern (BRAR) geleitet, welche die Ströme abfordern und weiter zu den einzelnen Benutzern verteilen.

Der rechte Teil der Abb. 1 zeigt die Benutzer-Seite (Home Area). Von hier aus gesehen gibt es neben dem Passiven Optischen Netzwerk (PON) auch die Digital Subscriber Line (DSL) - Technologie mit ihren unterschiedlichen Übertragungsgeschwindigkeiten, um die Daten aus

dem Netz zu erhalten. Mit Very High DSL (VDSL) kann dem Endanwender bis zu 25 Mbit/s bereitgestellt werden. In den Verteilerstationen der Netzanbieter befindet sich ein Digital Subscriber Line Access Multiplexer (DSLAM). Dieser Multiplexer ist mit der STB (Set Top Box) des Endverbrauchers (in der Praxis können auch mehrere vorkommen) direkt verbunden, die für die Decodierung des Datenstroms zum Fernsehsignal verantwortlich ist.

3 Protokollmodell für IPTV

Die Abb. 2 zeigt den gängigen Protokollstapel für IPTV.

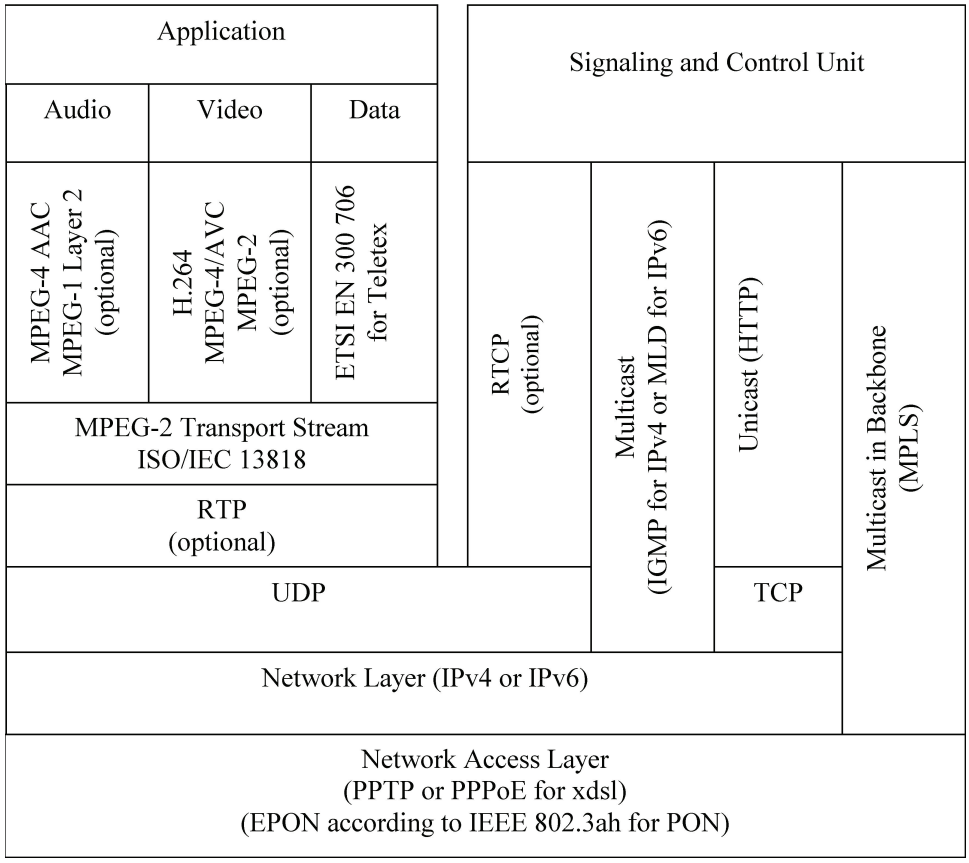


Abb. 2: Protokollstapel für IPTV

Die Übertragung der Audio-, Video- und Datenströme erfolgt verpackt im MPEG-2 Transportstrom [2] über das UDP, da eine Fehlerkorrektur die Qualität der Übertragung in der Regel nur verschlechtern würde. Zur Unterstützung der Echtzeitübertragung steht optional das Protokoll RTP zur Verfügung.

Die Signalisierung und Kontrolle wird normalerweise über Multicast-Protokolle (IGMP oder MLD) abgewickelt. Im Backbone können solche Kanäle zusätzlich mit Hilfe von MPLS gekapselt verwendet werden. Dies erhöht die Effizienz im Netz und trägt zur Sicherheit der übertragenen Daten bei. Im Head End verwendet man auch (z.B. zur Umschaltung der Programme) den Unicast-Verkehr. Dies wird über das bekannte HTTP-Protokoll abgewickelt.

4 Methoden zur Beurteilung der QoS bei IPTV

Es gibt prinzipiell zwei Gruppen von Methoden zur Beurteilung der Qualität eines Dienstes: a) subjektive und b) objektive. Im ersten Fall arbeitet man mit Gruppen von Beurteilungspersonen und als Ergebnis erhält man einen durchschnittlichen Meinungswert in der MOS (Mean Opinion Score)-Skala (vgl. Tabelle 1). Hier spricht man über QoE (Quality of Experience).

Tabelle 1: MOS Skala

MOS Werte	Dienstqualität
5	Ausgezeichnet
4	Gut
3	Ordentlich
2	Mäßig
1	Schlecht

Die objektiven Methoden können in zwei Untergruppen aufgeteilt werden: a) Beurteilung aufgrund der Netzwerkparameter und b) Beurteilung aufgrund der Referenzdateien unter Berücksichtigung der physiologischen Erkenntnisse des menschlichen Seh- und Hörvermögens. Eins ist den beiden Untergruppen gemeinsam: Die Beurteilung wird unter Verwendung von technischen Messsystemen objektiv und nachvollziehbar ausgeführt. Bei den Methoden mit Referenzdateien arbeitet man mit festgelegten, den einzelnen Diensten speziell angepassten Algorithmen. Ein Überblick über die bekanntesten Algorithmen zur Beurteilung der Qualität bei den sog. „Triple Play“ Diensten, d.h. Dienste vom Typ Sprache/Audio/Video/Daten, zeigt die Abb. 3. Hier spricht man über QoS (Quality of Service).

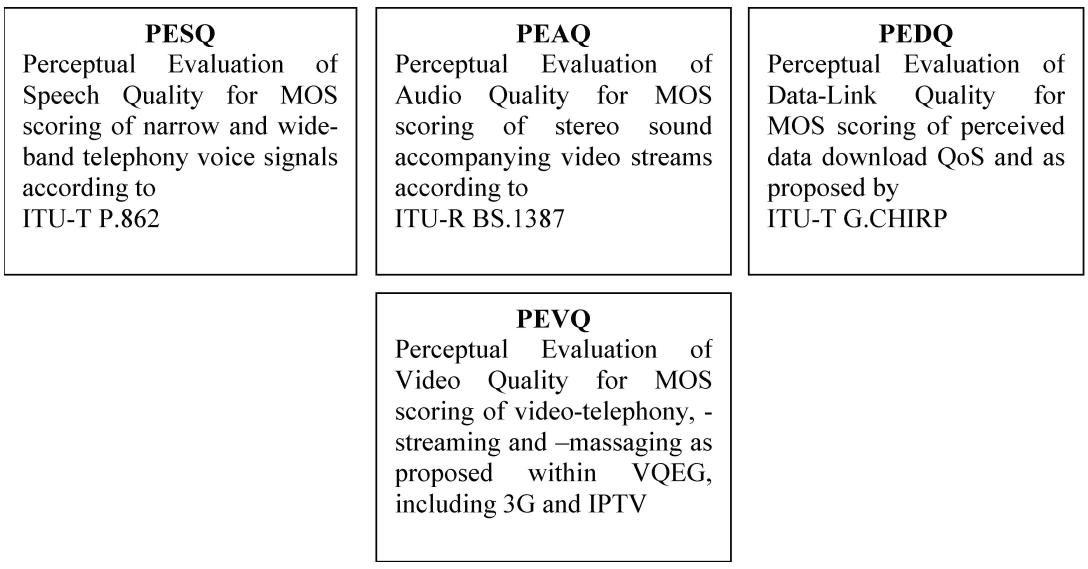


Abb. 3: Methoden zur Beurteilung der Qualität bei “Triple-Play”-Diensten

An dieser Stelle stellt sich die Frage, welche Abhängigkeit zwischen den QoE-Werten und den QoS-Werten existiert. Hat die Kurve zur Darstellung dieser Abhängigkeit einen linearen, konvexen oder konkaven Verlauf? Diese Frage kann zur Zeit nicht eindeutig beantwortet werden. Hier sind weitere Untersuchungen notwendig. Die erste Informationen über diese Abhängigkeit bezüglich des Sprachdienstes liefert die Spezifikation G.107 von ITU-T [3].

Dort wird als Maß für QoS der R-Faktor angenommen, der in dem E-Modell bei der objektiven Beurteilung der Qualität des Sprachdienstes das Ergebnis darstellt. Der R-Faktor kann Werte zwischen 0 und 100 annehmen. Die Abhängigkeit zwischen QoE und QoS bei dem Dienst wird anhand der folgenden Formel errechnet:

$$QoE\,[MOS] = 1 + 0{,}035R + 7R(R-60)(100-R)10^{-6}$$

Die Abb. 4 zeigt den gemäß der o.g. Formel ermittelten Verlauf der Abhängigkeit zwischen QoE und QoS beim Sprachdienst.

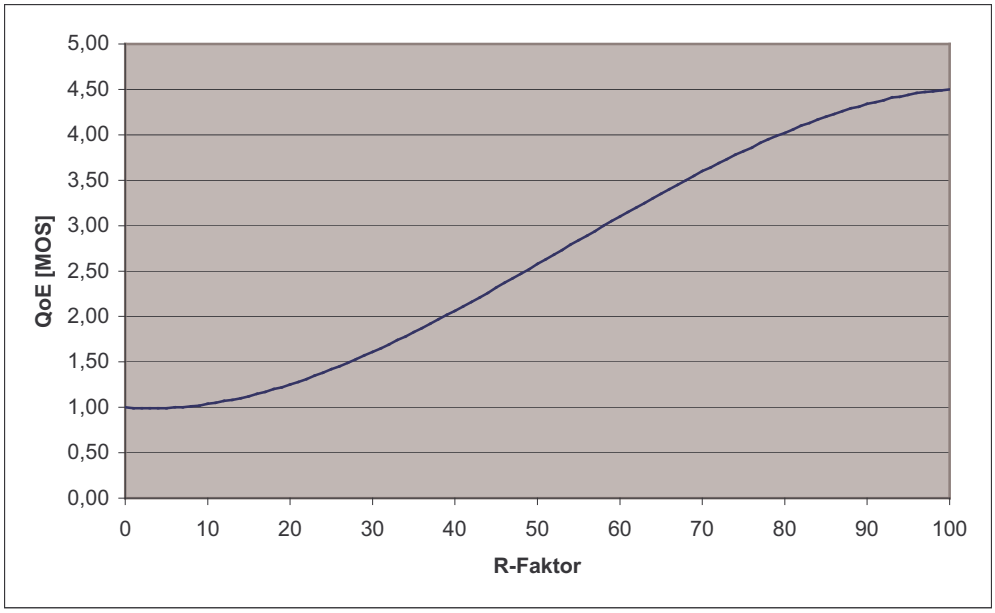


Abb. 4: QoE-Werte als Funktion des R-Faktors beim Sprachdienst

Man sieht, dass der Verlauf der Kurve nur im Bereich von 20 bis 80 etwa linear ist. In diesem Bereich stimmen also subjektive Wahrnehmungen mit objektiv ermittelter Qualität überein. Links und recht flacht die Kurve ab. Im unteren Bereich sieht man, dass sehr schlechte Qualität subjektiv oft positiver bewertet wird. Das ist damit zu begründen, dass die Testpersonen irgendwie versuchen, noch etwas von dem vermittelten Inhalt zu erkennen. Im oberen Bereich wird die Qualität oft schlechter bewertet. Das liegt daran, dass die Testpersonen in diesem Bereich immer etwas vorsichtiger bewerten und selbst bei optimaler Qualität keine Höchstpunkzahl vergeben.

Im Weiteren wird ein Überblick über die verschiedenen Möglichkeiten der Qualitätsbewertungen von Videoübertragungen via IP geben.

4.1 Quality of Service auf der Basis der Netzwerkleistung

Die Messung der QoS auf Basis der Netzwerkleistung erfolgt in Bezug auf folgende Parameter:

- *Bandbreite (Bandwidth),*
- *Paketverluste (Packet loss),*
- *Verzögerung (Delay),*
- *Jitter.*

Diese Parameter sind auf der Netzwerkebene zu messen, über die die eigentlichen Transportströme übertragen werden. Im Fall von IPTV sind diese in den Protokollen IP, UDP und RTP (Real Time Protocol) eingekapselt. Die Bestimmung der Paketverluste, Verzögerung und Jitter erfordert, dass die Datenströme via RTP übertragen werden, denn bei einer einfachen UDP-Übertragung fehlen die Mechanismen mit denen sich diese Parameter detektieren bzw. berechnen lassen. Das RTP-Protokoll liefert die dazu notwendigen Informationen. Die Empfehlung ITU-T Y 1540/1541 [4] und der Standard IETF REC 2330 [5] legen hier die Rahmen für die Bestimmung der QoS aufgrund der Netzwerkparameter fest.

4.2 Quality of Service auf Basis der Videoleistung

Die Messung der QoS auf Basis der Video-Leistung erfolgt in Bezug auf folgende Parameter:

- *Programmreferenzuhr Jitter (Program Clock Reference Jitter: PCR Jitter),*
- *Synchronisationsverlust (Synchronisation Loss),*
- *Kontinuitätsfehler (Continuity Error).*

Die Video-Daten werden bei IPTV zusammen mit Audio, Text und den Tabellen als MPEG-2-Transportstrom übertragen, der auf das UDP/RTP-Protokoll aufgesetzt wird. Aus diesem Transportstrom können die oben aufgeführten Parameter ermittelt werden. Der Standard ETSI TR 101-290 [6] beschreibt diese und weitere Parameter (auch für die physikalische Schicht) und deren Grenzwerte.

Wie man sich vorstellen kann, sind die Parameter aus den Abschnitten 4.1 und 4.2 nicht unabhängig voneinander. Verfügt die Übertragung nicht über die erforderliche Bandbreite oder ist die Netzverzögerung oder der Jitter zu groß, so wird auch der PCR-Jitter groß. Gehen Pakete verloren, kommt es zu Synchronisationsverlusten und Kontinuitätsfehlern. Das bedeutet, dass für eine entsprechende Dienstgüte sowohl die Netzleistung als auch die Videoleistung in Ordnung sein müssen.

Die Spezifikation TR 101-290 mit dem Originaltitel „Digital Video Broadcasting (DVB) – Measurement Guide Lines for DVB“ geht nicht explizit auf DVB-IPI ein. Das schränkt ihre Eignung für IPTV etwas ein. So werden einige Forderungen bei IPTV fast nicht erfüllt werden können. Da aber große Teile durchaus anzuwenden sind, wird die ETSI-Spezifikation allgemein als Grundlage für QoS-Messungen anerkannt und vielfach verwendet.

Innerhalb der Protokollebene definiert die Spezifikation TR 101-290 Parameter für die Beurteilung der Qualität des MPEG-2 Transportsstroms. Die zu ermittelnden Parameter sind in drei Prioritäten mit folgenden Inhalten aufgeteilt:

- **Erste Priorität**: Notwendig für die Decodierbarkeit (Basisüberwachung).
  Zu den festgelegten Parametern gehören hier: TS_sync_loss, Sync_byte_error, PAT_error, Continuity_count_error, PMT_error und PID_error.
- **Zweite Priorität**: Empfohlen für kontinuierliche oder periodische Überwachung.
  Zu den festgelegten Parametern gehören hier: Transport_error, PCR_repetition_error, PCR_discontinuity_error, PCR_accuracy_error, PTS_error und CAT_error.
- **Dritte Priorität**: Applikationsabhängige Überwachung.
  Zu den festgelegten Parametern gehören hier: NIT_actual_error, NIT_other_error, SI_repetition_error, Unreferenced_error, SDT_actual_error, SDT_other_error, EIT_actual_error, EIT_other_error, EIT_PF_error, RST_error und TDT_error.

4.3 PEVQ (Perceptual Evaluation of Video Quality)

Die Messung der Videoqualität nach PEVQ (stützt sich auf ITU-T J 144 [7]) bezeichnet man als „Full Reference (FR)“ -Messung, da für die Bestimmung des Qualitätswertes sowohl das Original-Video als auch das empfangene, beeinträchtigte Video benötigt wird.

PEVQ basiert auf der Nachbildung des menschlichen Sehsystems, mit dem die Wahrnehmung der Abweichungen im Videosignal ermittelt und quantitativ bestimmt werden. Eine Übersicht der Funktion von PEVQ zeigt die Abb. 5 [8].

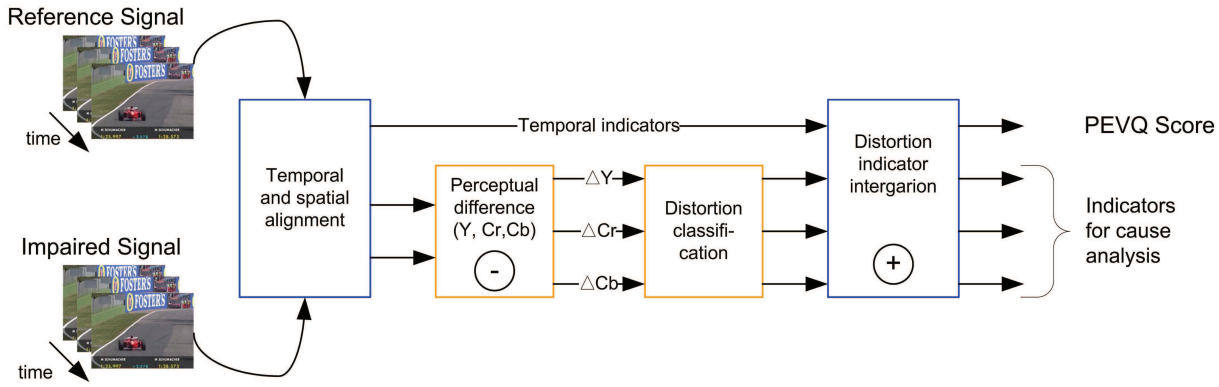


Abb. 5: Blockschaltbild von PEVQ

Der Algorithmus kann in vier separate Blöcke aufgeteilt werden:

I. Das Pre-Processing ist verantwortlich für den räumlichen und zeitlichen Abgleich des Original-Videos mit dem beeinträchtigten Video. Dieser Prozess ermöglicht, dass nur zueinander gehörende Bilder miteinander verglichen werden.
II. Der zweite Block errechnet die wahrgenommenen Unterschiede der beiden Signale. Es werden dabei wirklich nur die Unterschiede berücksichtigt, die vom menschlichen Zuschauer überhaupt bemerkt werden können. Außerdem liefert die Aktivität der Bewegung im Referenzsignal einen weiteren Wert, der die zeitlichen Informationen darstellt. Dieser Wert ist wichtig, weil er in Betracht zieht, dass bei einer Bilderserie mit geringer Aktivität die Wahrnehmung von Details sehr viel größer ist als in Bilderserien mit schnellen Bewegungen.
III. Der dritte Block stuft die vorher errechneten Werte ein und ermittelt bestimmte Arten von Verzerrungen.
IV. Schlussendlich werden im vierten Block alle entsprechenden Werte entsprechend der ermittelten Verzerrungen zusammengefasst, um das Ergebnis - einen MOS-Wert - zu bilden.

Außer dem MOS-Wert als endgültiges Qualitätsmaß werden bei PEVQ weitere Indikatoren für spätere Auswertungen der Ursache für die Qualitätsbeeinträchtigungen zur Verfügung gestellt:

- *Verzerrung (Distortion),*
- *Verzögerung (Delay),*
- *Helligkeit (Luminance),*
- *Kontrast (Contrast),*
- *PSNR (Peak Signal to Noise Ratio),*
- *Ruckeln (Jerking),*

- *Unschärfe (Blurring),*
- *Blockbildung (Block Construction),*
- *Überspringen von Bildern und eingefrorene Bilder (Jumping and Frozen of Picture),*
- *Effektive Bildrate (Effective Picture Rate),*
- *Zeitliche und räumliche Aktivität (Time and Areal Activity).*

Der PEVQ-Ansatz der Qualitätsbestimmung enthält sowohl die Auswirkungen im Paket-Bereich, wie Paketverluste und Jitter, als auch auf das Videosignal bezogene Beeinträchtigungen, wie Blockartefakte, Ruckeln, Unschärfe und Verzerrungen, die vom Codec verursacht werden.

Das Problem von PEVQ besteht darin, dass bis heute von den Algorithmen nur Auflösungen bis maximal VGA (640x400 Pixel) und PAL-Auflösung (768x576 Pixel) unterstützt werden. Da IPTV immer häufiger in HD-Auflösung (bis zu 1920x1080 Pixel) angeboten wird, wäre es sinnvoll, das Verfahren auf diesen neuen Service anzupassen. Laut Informationen des Lizenzhalters, der Firma Opticom [8], sollte dies in der zweiten Hälfte des Jahres 2007 erfolgen.

5 TraceView IPTV - Messsystem

Unter Verwendung von geeigneten Netzwerkprotokollanalysatoren können alle Parameter in den einzelnen Prioritätsklassen gemessen und ausgewertet werden. Dies wird weiter am Beispiel des Messsystems „TraceView IPTV“ der Firma ITD [9] (Kooperationspartner der FH Flensburg) verdeutlicht. Das Messsystem kann auch zum Zweck der Langzeitüberwachung von IPTV-Verbindungen eingesetzt werden. Hierbei soll das Tool „Trafficlyser IPTV Monitor“ [9] eingesetzt werden. Die Abb. 6 zeigt beispielhaft die Bedienoberfläche dieses Messsystems mit einem Ausschnitt aus einer realen IPTV-Messung.

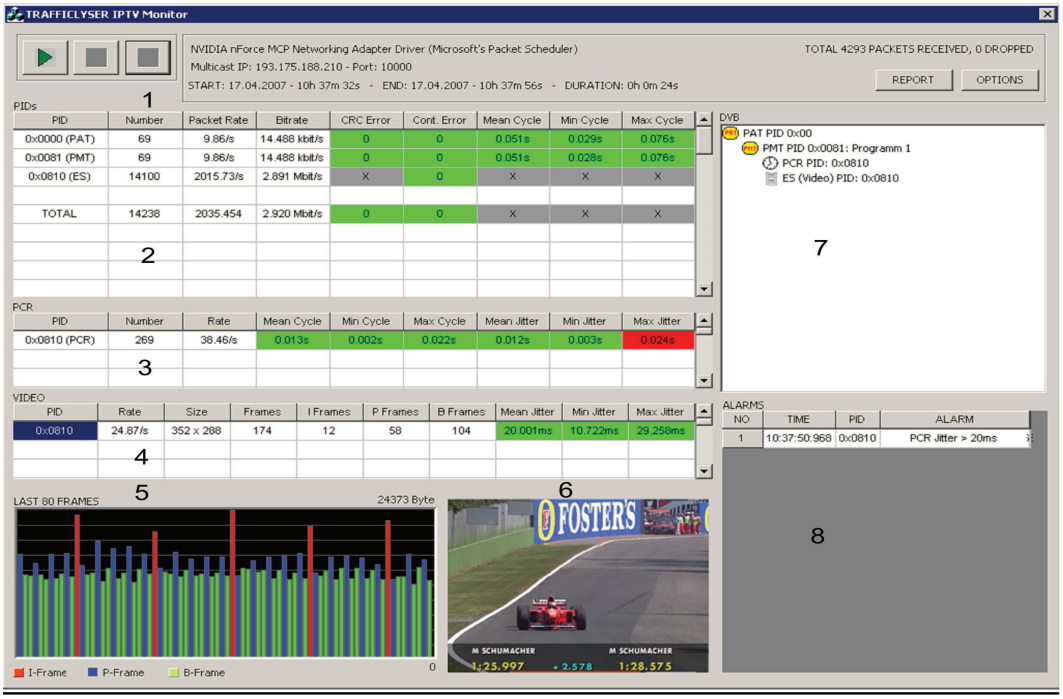


Abb. 6: Bedienoberfläche des Messsystems „Trafficlyser IPTV Monitor“ in einem Beispiel

Das Feld (1) zeigt die allgemeinen Informationen zum Messsystem. Zu denen gehören: Art des ausgewählten Netzwerkadapters, überwachte Multicast-Adresse und Port, Start- und Endzeit sowie Dauer der Aufzeichnung, Anzahl der insgesamt empfangenen Pakete sowie Anzahl der verlorenen Pakete. Das Feld (2) enthält Informationen über Anzahl von MPEG-2-Transportpaketen, Paket- und Bitrate, CRC- und Kontinuitätsfehler sowie Tabellenzyklen. Im Feld (3) werden alle auftretenden PCR (Program Clock References) aufgelistet. Dabei werden Anzahl, Rate, Zyklen und Jitter berücksichtigt. Das Feld (4) enthält die Videoliste. Hier werden Anzahl und Art der Frames, Bildgröße, Rate, Jitter ausgegeben. Das Feld (5) enthält das Framediagramm. Hier werden Frameart und -größe des in der Videoliste markierten Elementarstroms graphisch dargestellt. Im Feld (6) können die I-Frames des in der Videoliste markierten Elementarstroms decodiert und ausgegeben werden (nur bei unverschlüsselten Elementarströmen). Das Feld (7) zeigt die Struktur der DVB-Übertragung mit allen relevanten PIDs (Program Identifier). Die Darstellung weist eine Baumstruktur auf. Das Feld (8) beinhaltet die Alarmliste. Hier werden auftretende Alarme (PCR- und Tabellenzyklen, CRC- und Kontinuitätsfehler sowie PCR- und Frame Jitter-Überschreitungen) chronologisch aufgelistet.

6 Messungen in einer realen Umgebung

In diesem Kapitel soll die Eignung der IP-Transportplattform für den neuen Multimediadienst IPTV überprüft werden. Für diesen Zweck wird die in der Abb. 7 dargestellte Messumgebung aufgebaut und zu Messzwecken genutzt.

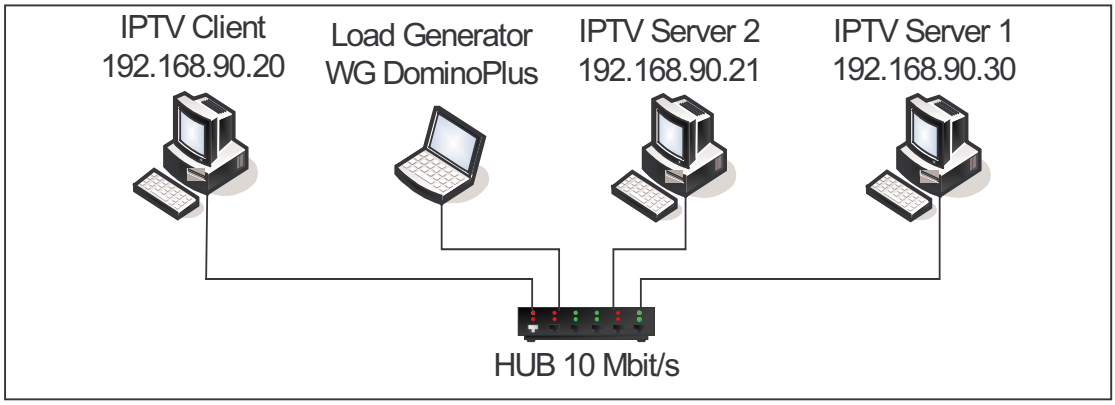


Abb. 7: Verwendete Messumgebung

An einem 10 Mbit/s Hub werden zwei IPTV Server (zur Emulation der IPTV-Sessions), ein IPTV Client (zur Messung der QoS) und ein Lastgenerator (zur Erzeugung der Hintergrundlast) angeschlossen. Die beiden IPTV-Server erzeugen und verwenden einen MPEQ-2 Transportstrom mit in ihm verankerten Video-Sequenzen (Referenzvideo codiert nach H.264 = MPEG-4/AVC [10]). Auf dem IPTV-Client ist die Software „TraceView IPTV“ installiert. Das Messsystem ist in der Lage, sowohl die Netzwerkparameter zu messen als auch die QoS bei IPTV nach dem PEVQ-Algorithmus zu beurteilen. Die Hintergrundlast wird von 0 bis zu 50 % (bezogen auf die 10 Mbit/s) verändert. Dazu werden Ethernetrahmen mit unterschiedlichen Längen (von 64 bis 1500 Bytes) und Zwischenankunftszeiten gemäß einer negativ-exponentiellen Verteilung generiert und ins Intranet eingespeist. Die erhaltenen Ergebnisse aus dieser Untersuchung sind in der Tabelle 2 und in der Abb. 8 dargestellt.

Tabelle 2: Messergebnisse aus der Untersuchung

Messung Nr.	Generierte Hintergrundlast [%]	PEVQ-Wert [MOS]	Bemerkungen
1	0	4,184	
2	10	4,184	
3	20	4,184	
4	22	3,874	
5	24	3,415	
6	26	3,161	
7	28	3,028	
8	30	3,028	
9	35	3,028	
10	40	1	N.A.
11	50	1	N.A.

Die Ergebnisse aus der Tabelle 2 und aus der Abb. 8 zeigen deutlich, dass die Hintergrundlast einen sehr großen Einfluss auf die Qualität des Dienstes IPTV hat. Man sieht, dass die QoS ab der Grenze 35 % der Hintergrundlast sich wesentlich verschlechtert. Ab dieser Grenze können keine akzeptablen IPTV-Übertragungen stattfinden. Paketverluste steigen in diesem Bereich sehr schnell und auch viele andere für Videoübertragung wichtige Parameter (vgl. Kapitel 4.3) können nicht angehalten werden. Grund dafür ist die fehlende Bandbreite im Hintergrund. Eine Standard IPTV-Sendung braucht eine Bandbreite von ca. 3 Mbit/s. Unter der Berücksichtigung dieses Wertes ist es klar, warum bei der Hintergrundlast von ca. 35 % und bei zwei in der verwendeten IP-Plattform (10BASE-T) gleichzeitig aktiven IPTV-Servern keine fehlerfreie IPTV-Sendung zu Stande kommen konnte. Die Kollisionsanzahl für die Ethernetrahmen ist einfach zu groß und die daraus resultierenden Variationen der Zwischenankunftszeiten sind für den zeittreuen Dienst wie IPTV nicht zu akzeptieren. Dazu kommen noch die Ethernetrahmenverluste. Oft werden Ethernetrahmen mit I-Frames als Inhalt verloren gehen, was natürlich richtige Bilddecodierung unmöglich macht. QoS und QoE sinken in diesem Fall rapide.

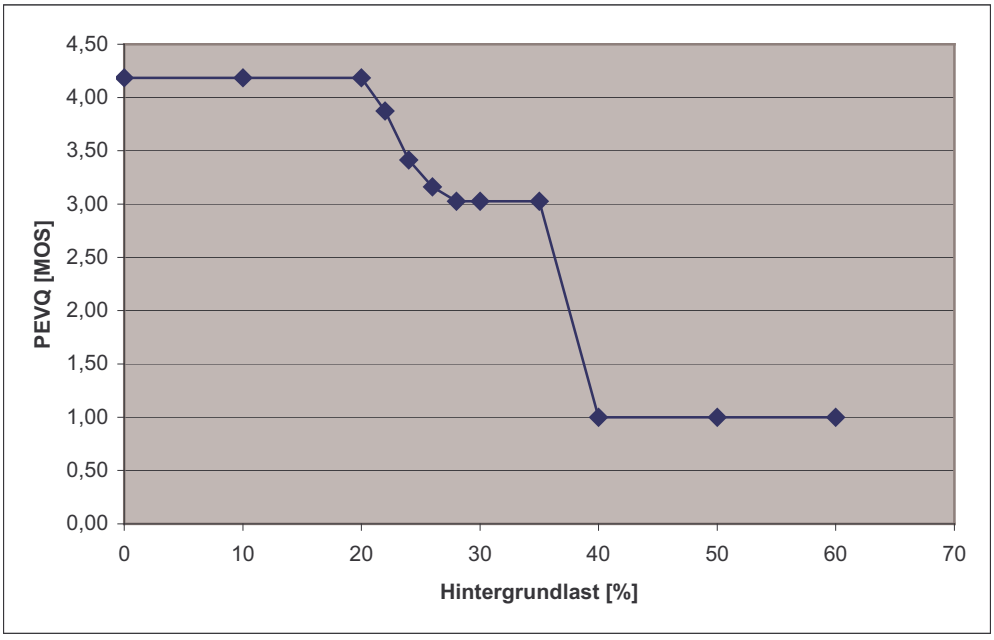


Abb. 8: Graphische Darstellung der Messergebnisse

Fast man die Erkenntnisse der im Rahmen dieser Arbeit durchgeführten Untersuchungen zusammen und berücksichtigt weitere reale Messreihen aus hier nicht vorgestellten IPTV-Umgebungen, dann lässt sich folgendes sagen:

- Die Signalisierung bei IPTV nutzt eine Unicast-Verbindung und verwendet dabei entweder das IGMP-Protokoll oder das HTTP-Protokoll.
- IPTV-Kommunikation selbst nutzt eine Multicast-Verbindung im Simplex-Betrieb. Die dazu benötigten Multicast-Adressen werden durch Serviceprovider festgelegt.
- Bei IPTV-Multicast-Kommunikation ist der Verkehr vom Typ CBR.
- Eine SDTV IPTV-Multicast-Kommunikation nach H.264 = MPEG-4/AVC nimmt ca. 3 Mbit/s Bandbreite in Anspruch.
- Eine HDTV IPTV-Multicast-Kommunikation nach H.264 = MPEG-4 /AVC nimmt ca. 10 Mbit/s Bandbreite in Anspruch.
- Bei IPTV werden die Nutzdaten aus der Applikationsschicht in dem Transportstrom für den Standard MPEG-2 übertragen. Ein MPEG-2-Transportpaket weist die konstante Länge von 188 Bytes auf.
- 7 MPEG-2-Transportpakete werden zu einer Gruppe zusammengefasst und entweder in einem UDP- oder in einem RTP-Paket (optional) übertragen. Daraus resultiert die konstante Länge von ca. 1350 Bytes für diese Pakete. Bei Messungen im ersten Fall muss auf die Multicast-Adresse gefiltert werden. Im zweiten Fall kann auch zusätzlich auf die aufgebauten RTP-Sessions zugegriffen werden.
- Bei einer Hindergrundlast von 0% – 35 % (bezogen auf 10BASE-T) und bei zwei gleichzeitig aktiven TV-Sendungen ist der Dienst IPTV stabil.
- Bei einer Hindergrundlast größer als 35 % verschlechtert sich die Kommunikation wesentlich bis sie letztendlich nicht mehr möglich ist.
- Längere Ethernetrahmen im Hintergrund begünstigen die IPTV-Übertragungen im Vergleich zu kürzeren Rahmen.
- Soll die Effektivität der Übertragung erhöht werden (um z.B. Jitter zu kompensieren), muss die Kapazität des Ausgleichspuffers entsprechend dimensioniert werden.

Unter Berücksichtigung der o.g. Feststellungen lässt sich bei der Multimediakommunikation in einer genutzten IP-Umgebung folgende Faustregel für die Dimensionierung angeben:

Die im Netz zur Verfügung stehende Bandbreite(BB) muss mindestens so groß sein wie die benötigte Gesamtbandbreite.

Benötigte Gesamtbandbreite = 1,33 x (Min BB für Sprache + Min BB für Audio + + Min BB für Video + Min BB für Daten).

Der Faktor 1,33 berücksichtigt den zusätzlichen Verwaltungsverkehr in der IP-Transportplattform. Bei der Bestimmung der minimalen Bandbreite pro Dienst muss die Anzahl der für den Dienst vorgesehenen Clients berücksichtigt werden. Oder umgekehrt, ausgehend aus der im Netz zur Verfügung stehenden Bandbreite (Sie kann durch Messungen ermittelt werden.) lässt sich die maximale Anzahl der Clients pro Dienst ermitteln. Diese 133%-Regel wurde in vielen Experimenten angewandt und hat sich als sehr geeignet für die Praxis erwiesen.

7 Zusammenfassung

Im Rahmen dieser Arbeit wurden theoretische und praktische Aspekte bei dem neuen Multimediadienst IPTV erörtert. Es lässt sich sagen, dass die Technik für diesen neuen Dienst

vorhanden ist. Dies bestätigen in der Praxis erste Implementierungen und Angebote auf dem Gebiet IPTV [11], die von mehreren Serviceprovidern in verschiedenen Staaten angeboten werden.

Betrachtet man die typische Struktur für IPTV, dann muss man feststellen, dass in allen Bereichen dieser Struktur, d.h. Head End, IP-Multicast-Network and Home Area die technischen Probleme weitgehend gelöst sind. Die im Head End vorkommenden Geräte, z.B. MPEG-4/AVC-Codierer/Decodierer, MPEG-2-Multiplexer, u.a. funktionieren gut und sind auf dem Markt erhältlich. Auch die IP-Netzinfrastruktur stellt ein ausgereiftes und erprobtes Produkt dar. In dem IP-Transportsystem können mehrere standardisierte Mechanismen zur Erhöhung der QoS eingesetzt werden. Zu den wichtigsten Mechanismen gehören hier: Verwendung von Prioritäten (IEEE 802.1p/Q), Link Aggregation (IEEE 802.3ad), MPLS (Multi-Protocol Label Switching; IETF RFC 3031), Differentiated Services (IETF RFC 791 und RFC 2474) und RSVP (Resource Reservation Protocol; IETF REC 2205). Die Home Area kann unter Verwendung von heutzutage leistungsfähigen Mitteln wie DSL oder PON an das IPTV-Netz angeschlossen werden. Die der Home Area vorkommenden Geräte wie DSLAM, OLT (Optical Line Terminal), ONT (Optical Network Terminal), STB (Set Top Box), TV-Geräte sind ebenfalls ausgereifte Produkte.

Ein Dauerbrenner bei IPTV war und ist die Quality of Service (QoS). Wie in dieser Arbeit gezeigt, kann die QoS auf verschiedene Weise durch spezialisierte Messsysteme festgestellt werden. An dieser Stelle muss man jedoch unterstreichen, dass es zur Zeit auf dem Markt nicht so viele und vor allem kaum preiswerte Messsysteme zur Bestimmung der QoS bei IPTV gibt. Hier sind weitere Arbeiten notwendig, um vielfältige und leistungsfähige Messsysteme für QoS zu entwickeln. Die Aufweitung des existierenden Algorithmus gemäß PEVQ auf die HDTV-Version sollte schnell erfolgen, um in Zukunft objektiv die Qualität des Dienstes IPTV im vollen Umfang beurteilen zu können. QoS und QoE treffen sich in der Home Area. Die letzte Meile stellt zur Zeit einen Engpass dar. Hier ist zu erwarten, und erste Messungen bestätigen dies, dass die QoS dort besonders beeinträchtigt wird. Dies sollte schellst möglich untersucht werden, um später geeignete Gegenmaßnahmen ergreifen zu können. Eine Hilfestellung für diesen Bereich kann die im Rahmen dieser Arbeit vorgestellte Dimensionierungsregel geben.

8 Quellen

[1] Scherer, Ch.: Architekturkonzept T-Home-Speed. T-Online International AG (Editor), 2005 (internal Paper)
[2] MPEG-2 Transport Stream: http://erg.abdn.ac.uk/research/future-net/digital-video/mpeg2-trans.html
[3] ITU-T REC G.107: http://www.itu.int/rec/T-REC-G/e
[4] ITU-T REC 1540/1541: http://www.itu.int/rec/T-REC-Y/e
[5] IETF REC 2300: http://www1.tools.ietf.org/html/rfc2300
[6] ETSI REC TR 101290: http://telin.ugent.be/~kayzlat/DVB-H/tr101290.v1.2.1.pdf
[7] ITU-T REC J 144: http://www.itu.int/rec/T-REC-J/e
[8] PEVQ: http://www.opticom.de/download/PEVQ-WP-v07-A4.pdf
[9] TraceView IPTV : http://www.trafficlyser.de
[10] Symes, P.: Digital Video Compression. McGraw-Hill Professional, New York, 2004
[11] Fischer, M.: Drei gewinnt. Funkschau, no 5/2005, pp. 28-30

Untersuchung der Fehlerrobustheit H.264/AVC-kodierter Videoströme bei simulierten Übertragungsverlusten

Michael Kiritz

Arbeitsgruppe Telekommunikation und Rechnernetze
Department Informatik, Universität Hamburg

Abstract: Der aktuelle Videokodierungsstandard H.264/AVC stellt eine Reihe von Techniken zur Verfügung, welche die Auswirkungen von durch Übertragungsverluste verursachten Bildfehlern minimieren sollen. In diesem Beitrag werden die Auswirkungen von einigen dieser Techniken auf die durch die kodierten Videoströme erzeugten Lasten sowie auf die Fehlerrobustheit der kodierten Videoströme untersucht. Es wird gezeigt, dass die Fehlerrobustheit durch zusätzlich erzwungene Intrakodierung deutlich verbessert werden kann, wobei die Anzahl der intrakodierten Makroblöcke einen größeren Einfluss hat als deren Anordnung. Zudem zeigen die Untersuchungen für die Verwendung von flexibel angeordneten I-Makroblöcken innerhalb eines Bildes eine Verbesserung der Fehlerrobustheit bei Streuverlusten.

Keywords: Videokommunikation, Videoströme, Lastcharakterisierung, Fehlerrobustheit, Bildqualität

1 Einleitung

Übertragungen von Videoinhalten sind wie alle Datenübertragungen anfällig für Störungen. Die Behandlung von Übertragungsfehlern ist sehr wichtig, da Anwender bei Videoinhalten, je nach Anwendung, u.U. sehr empfindlich auf Qualitätseinbußen reagieren. Eine möglichst gute Qualität der Videoinhalte ist eine der wichtigsten Voraussetzungen für die Akzeptanz und Nutzung neuer digitaler Medien. Insbesondere im mobilen Bereich wird ein starkes Wachstum bei der Übertragung von Videoinhalten erwartet. Gleichzeitig ist gerade die Mobilkommunikation über drahtlose Netze besonders anfällig für Übertragungsfehler.
Wenn eine Videokommunikation in Echtzeit ablaufen soll, sind die Möglichkeiten zur Fehlerbehandling stark eingeschränkt. Es ist daher besonders wichtig, bei einer solchen Kommunikation die Daten so zu kodieren, dass Übertragungsfehler nur möglichst geringe Auswirkungen auf die vom Anwender wahrgenommene Qualität haben. Es gibt eine Reihe von Techniken, um die Robustheit von Videoinhalten gegenüber Fehlern zu verbessern. Im Folgenden werden einige dieser Techniken vorgestellt und untersucht, wobei sowohl die Auswirkungen auf die Fehlerrobustheit als auch auf die Laststruktur der kodierten Videoströme betrachtet werden.

Der aktuelle Videokodierungsstandard H.264/AVC [ITU05], welcher von den Expertengruppen VCEG (ITU-T) und MPEG (ISO/IEC) gemeinsam im JVT *(Joint Video Team)* entwickelt wurde, stellt u.a. für die Videokommunikation in Echtzeit geeignete Kodierwerkzeuge zur Verfügung. Es gibt verschiedene Profile, welche die Auswahl und Optimierung der vorhandenen Kodierwerkzeuge für verschiedene Anwendungsszenarien ermöglichen. Es wurden in H.264/AVC im Wesentlichen bereits in früheren Standards enthaltene Kodiertechniken aufgenommen, die komplett überarbeitet und optimiert wurden. Ziel war es, die Kodiereffizienz im Vergleich zu früheren Standards deutlich zu verbessern. Die erweiterten und flexibleren Kodiertechniken erfordern jedoch aufwendige Optimierungsverfahren, welche den Ressourcenbedarf von H.264/AVC gegenüber älteren Verfahren deutlich erhöhen.
Zur Verbesserung der Kodiereffizienz verwenden Videokodierungsstandards räumliche und zeitliche Vorhersagetechniken. Die intensivere Nutzung solcher Techniken bei H.264/AVC erhöht die Gefahr der Ausbreitung von Kodier- und insbesondere Übertragungsfehlern. Um dieser Gefahr zu begegnen, existieren in H.264/AVC im Vergleich zu früheren Standards

zusätzliche Techniken zur Verbesserung der Robustheit gegenüber Fehlern sowie der schnelleren Erholung von Fehlern.
Bereits in [Koh00] wurden die Auswirkungen von Übertragungsfehlern auf Videos untersucht, die mit den älteren Standards H.261 bzw. H.263 kodiert wurden. Ein Modell für die Fehlerakkumulation sowie ein Fehlertoleranzverfahren für H.261 bzw. H.263 wurden in [HKS+01] vorgestellt. In [Nor05] wurden die Auswirkungen zahlreicher Kodiertechniken, die der Verbesserung der Kodiereffizienz im Vergleich zu älteren Videokodierungsstandards dienen, auf die Laststruktur von H.264/AVC untersucht. Außerdem wurde die Gültigkeit eines in früheren Arbeiten entwickelten Lastmodells für die hybride Videokodierung (vgl. [Koh00][Zad01][ZH01]) generell für H.264/AVC nachgewiesen.

2 Fehlerrobustheitstechniken für H.264/AVC

In diesem Abschnitt werden die untersuchten Techniken kurz vorgestellt. Eine Darstellung der Methoden der Videokodierung findet sich z.B. in [WSBL03] oder ausführlicher in [Ric03]. Ein Überblick über alle für H.264/AVC vorgesehenen Fehlerrobustheitstechniken wird u.a. in [Wen03] gegeben.
Bei der Videokodierung werden die Bilder des zu kodierenden Videos in Makroblöcke (MB) unterteilt, welche jeweils 16 x 16 Bildpunkte groß sind. MB können zu Bildscheiben zusammengefasst werden. Ein Bild kann so in eine oder mehrere Bildscheiben aufgeteilt werden, die jeweils einen oder mehrere MB enthalten und weitgehend unabhängig voneinander kodiert bzw. dekodiert werden.
Es lassen sich prinzipiell zwei Arten von kodierten Bildern (Bildscheiben) unterscheiden: Die mit ausschließlich räumlicher und die mit vorwiegend zeitlicher Kodierung. Räumlich kodierte Bilder (Bildscheiben) werden auch als intrakodierte Bilder oder I-Bilder bezeichnet und enthalten ausschließlich intrakodierte Makroblöcke. Zeitlich kodierte Bilder (Bildscheiben) werden als interkodierte Bilder oder P-Bilder bezeichnet und können sowohl inter- als auch intrakodierte Makroblöcke enthalten.
Da sich mithilfe der zeitlichen Kodierung im Allgemeinen eine bessere Kompression erreichen lässt, werden die meisten Bilder eines Videos in der Regel als P-Bilder kodiert. Die Ähnlichkeiten zwischen den Bildern werden dabei ausgenutzt, indem für ein Bild nicht seine absoluten Abtastwerte sondern stattdessen Differenzwerte zu einem vorher kodierten Referenzbild kodiert werden. Diese Art der Kodierung hat aber auch ihre Nachteile, besonders wenn es zu Fehlern in einem Bild, z.B. durch Paketverluste während einer Übertragung, kommt. Wenn ein P-Bild, das als Referenzbild genutzt wird, fehlerhaft ist, können auch die nachfolgenden Bilder nicht korrekt dekodiert werden. Besonders bei Videos mit nur wenigen Veränderungen in der Szene können Fehler in nur einem Bild zu Folgefehlern in vielen weiteren Bildern führen. Um die Ausbreitung solcher Fehler zu begrenzen und um in einem Video Resynchronisationspunkte zu haben, die es ermöglichen zu verschiedenen Stellen in dem Video zu springen, werden im Allgemeinen in regelmäßigen Abständen I-Bilder kodiert. I-Bilder können unabhängig dekodiert werden, da sie keine Referenzen auf vorherige Bilder nutzen.
Es gibt für H.264/AVC eine Reihe von Techniken zur Verbesserung der Robustheit des kodierten Videostroms gegenüber aus Übertragungsverlusten resultierenden Bildfehlern, insbesondere um die Ausbreitung von Bildfehlern über mehrere P-Bilder hinweg zu reduzieren auch ohne regelmäßige I-Bilder zu verwenden.

Für die Kodierung der Videos wurde der Referenzcodec jm11.0 des JVT [WWW1] verwendet. In dieser Version waren noch nicht alle in [Wen03] beschrieben Techniken vollständig und fehlerfrei implementiert. Es wurde das Video ***foreman*** im QCIF-Format (176x144 Bildpunkte bzw. 11x9 MB) [WWW2] verwendet, wobei jeweils 380 Bilder bei 30

Bildern pro Sekunde kodiert wurden. Eine vollständige Übersicht der übrigen verwendeten Parameter findet sich in [Kir06].
Es wurden Kodierungen mit folgenden Konfigurationen durchgeführt:

1. IP+:
   Nur das erste Bild wurde als I-Bild kodiert und alle übrigen Bilder als P-Bilder. Diese Kodierkonfiguration weist keinerlei Fehlerrobustheit auf und dient als Vergleichsbasis für die übrigen Kodierungen.
2. IP(9):
   Jedes neunte Bild wurde als I-Bild kodiert und alle übrigen als P-Bilder. Es gibt bei H.264/AVC auch die Möglichkeit spezielle I-Bilder, so genannte IDR-Bilder *(Instantaneous Decoder Refresh)*, zu verwenden. Der Kodierer kann bei der Kodierung von P-Bildern nicht über IDR-Bilder hinweg referenzieren. IDR-Bilder konnten mit jm11.0 jedoch nicht verwendet werden, da die Dekodierung von verlustbehafteten Videoströmen dann häufig mit einer Fehlermeldung abbrach.
3. IP+ & I-MB-Line:
   Wie bei 1, aber in jedem P-Bild wurde die Kodierung einer Makroblockzeile als 11 I-MB erzwungen. Nach neun Bildern wurde somit jede Zeile einmal als I-MB-Zeile kodiert.
4. IP+ & I-MB 11:
   Wie bei 1, aber in jedem P-Bild wurde die Kodierung von 11 zufällig angeordneten I-MB erzwungen.
5. IP+ & FMO1:
   Wie bei 1, aber die MB jedes Bildes wurden in der Art eines Schachbrettmusters in zwei Bildscheibengruppen aufgeteilt. FMO *(Flexible Macroblock Ordering)* bietet die Möglichkeit, die MB eines Bildes flexibel auf mehrere Bildscheibengruppen aufzuteilen.
6. IP+ & FMO1 & I-MB-Line:
   Eine Kombination aus 3 und 5.
7. IP+ & FMO1 & I-MB 11:
   Eine Kombination aus 4 und 5.
8. IP+ & FMO1 & I-MB 33:
   Wie 7, aber statt mit 11 mit 33 zufällig angeordneten I-MB pro P-Bild.

3 Datei- und Bildrahmenlängen

Bereits in [Nor05] und [Nor06] wurden die Auswirkungen von verschiedenen Kodiertechniken zur Verbesserung der Kodiereffizienz von H.264/AVC auf die Charakteristiken der Bildrahmenlängen sowie der erzeugten Lasten dargestellt. In diesem Abschnitt werden die Auswirkungen der verschiedenen Kodiertechniken zur Verbesserung der Fehlerrobustheit bei ansonsten jeweils identischen Kodierparametern auf die Dateilängen der kodierten Videos sowie auf die Charakteristiken der Bildrahmenlängen beispielhaft für das Video ***foreman*** vorgestellt. In [Kir06] sind darüber hinaus die Auswirkungen für zwei weitere Videos zu finden, welche andere Charakteristiken aufweisen.

In Tabelle 1 werden die Gesamtlängen der kodierten Videos sowie die Anzahl der RTP-Pakete dargestellt. Die zweite Spalte enthält die Dateilängen, so wie sie von der NAL an das Transportsystem übergeben werden. Diese stellen die Gesamtlast dar, welche von der Anwendungsschicht an die Transportschicht während des Übertragungszeitraums insgesamt übergeben wird. Diese Last beinhaltet sowohl die Daten der kodierten Bilder und Parametermengen als auch den RTP-Protokolloverhead aller Pakete. Die Prozentangaben in den Tabellen dienen jeweils zum Vergleich der Dateilänge bzw. Paketanzahl der Kodierkonfigurationen 2 bis 8 mit den Werten der IP+-Kodierung.

Kodierkonfiguration	Dateilänge [Byte]	Differenz zu IP+ [%]	Anzahl RTP-Pakete	Differenz zu IP+ [%]
1. IP+	797577	0	754	0
2. IP(9)	1003150	25,77	899	19,23
3. IP+ & I-MB-Line	1001849	25,61	860	14,06
4. IP+ & I-MB 11	1028052	28,90	878	16,45
5. IP+ & FMO1	818913	2,68	841	11,54
6. IP+ & FMO1 & I-MB-Line	1052671	31,98	1018	35,01
7. IP+ & FMO1 & I-MB 11	1052055	31,91	1014	34,48
8. IP+ & FMO1 & I-MB 33	1517798	90,30	1475	95,62

Tabelle 1: Dateilängen und Anzahl der Pakete für foreman

Die Kodierkonfigurationen 2, 3, 4, 6 und 7 erzwingen zur Verbesserung der Fehlerrobustheit jeweils die Intrakodierung von 11,11% der Makroblöcke eines Videos. Die daraus resultierende Vergrößerung der Dateilängen liegt dabei zwischen 25,61% und 31,98%. Die etwas größeren Werte bei den Kodierkonfigurationen 4 und insbesondere 6 und 7 im Vergleich zu 2 und 3 sind darauf zurückzuführen, dass hierbei durch die zufällige Verteilung den I-Makroblöcke bzw. die Aufteilung in zwei Bildscheibengruppen durch FMO1 keine intrakodierten Referenzpixel aus benachbarten Makroblöcken für die Intrakodierung genutzt werden können. Die Verdreifachung der zusätzlichen I-Makroblöcke bei Kodierkonfiguration 8 im Vergleich zu 7 führt ziemlich genau auch zu einer Verdreifachung der Dateilänge.
An der Anzahl der Pakete lässt sich erkennen, dass bereits bei der IP+-Kodierung fast alle Bilder zwei Pakete benötigen, wobei das zweite Paket häufig nur zum Teil ausgenutzt wird. Die weitere Vergrößerung der Anzahl der Pakete fällt deshalb bei den Kodierkonfigurationen 2 bis 4 prozentual etwas kleiner aus als die Vergrößerung der Dateilängen. Bei den Kodierkonfigurationen 5 bis 8 hingegen ist die prozentuale Vergrößerung der Anzahl der Pakete größer als bei den Dateilängen. Hier werden durch FMO1 zwei Bildscheibengruppen pro Bild erzwungen, wobei zum Teil offenbar mehrere Bildscheiben pro Bildscheibengruppe verwendet werden.

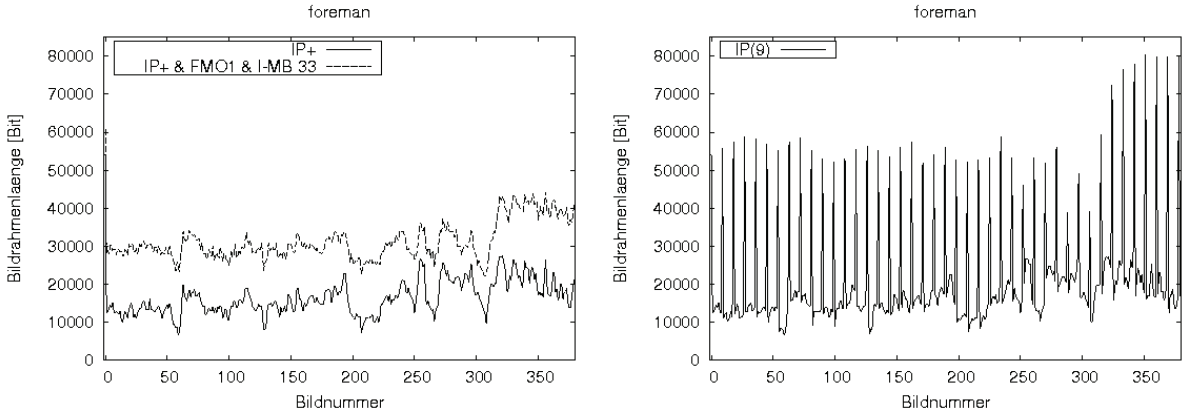


Abbildung 1: Ereignisspuren für Bildrahmenlängen von foreman

Abbildung 1 zeigt die Ereignisspuren der Bildrahmenlängen von ***foreman*** für drei ausgewählte Kodierkonfigurationen. Ausgewählt wurden die Spuren für die kleinste und größte Dateilänge, also für IP+ und IP+ & FMO1 & I-MB 33, sowie die Spur mit dem Verlauf von IP(9), welcher sich am deutlichsten von den anderen unterscheidet.

Die Vergrößerung der Bildrahmenlängen bei Intrakodierung von 33,33% der Makroblöcke ist deutlich zu erkennen. Auch bei Verwendung von FMO1 und I-MB 33 bleibt der generelle Charakter der jeweiligen Ereignisspuren erhalten, wobei die Schwankungen der Bildrahmenlängen bei den P-Bildern abnehmen. Diese Schwankungen bei der IP+-Kodierung sind darauf zurückzuführen, dass die Länge kleiner Bildrahmen durch einzelne Kodierentscheidungen deutlich beeinflusst werden. Da bei der Kodierung von I-Makroblöcken deutlich weniger Kodierentscheidungen getroffen werden können als bei P-Makroblöcken, ist anzunehmen, dass diese Schwankungen umso geringer ausfallen je höher der Anteil an I-Makroblöcken in P-Bildern ist.
Bei den Ereignisspuren für die IP(9)-Kodierung in Abbildung 5.2 sind die größeren Bildrahmenlängen der I-Bilder deutlich zu erkennen. Diese sind um einen Faktor von ca. 3 bis 7 größer als die P-Bilder. Aufgrund der ungleichmäßigen Lastverteilung eignet sich die IP(9)-Kodierung nur schlecht für eine Übertragung über Kanäle mit begrenzten Datenraten, da hierbei alle neun Bilder deutlich mehr Daten übertragen werden müssen als in der übrigen Zeit.

4 Auswirkungen von simulierten Übertragungsverlusten

In diesem Abschnitt werden die Auswirkungen von simulierten Übertragungsverlusten auf die Bildqualität von ***foreman*** bei Verwendung von Kodierkonfigurationen mit verschiedenen Fehlerrobustheitstechniken dargestellt. Die Auswirkungen werden jeweils anhand von Beispielen durch Graphen mit Δ-PSNR-Verläufen *(Peak Signal to Noise Ratio)* und Tabellen mit statistischen Daten verdeutlicht. Die Δ-PSNR-Werte wurden dabei als Differenz zwischen einem dekodierten Video mit und ohne Verluste berechnet. Größere Δ-PSNR-Werte stehen hierbei für eine schlechtere Bildqualität der Bilder des von Verlusten betroffenen Videos. Die Δ-A-PSNR-Werte wurden berechnet als das arithmetische Mittel der Δ-PSNR-Werte für alle Bilder eines Videos.
Es gibt Bemühungen, neue Verfahren zur Beurteilung der Bildqualität bzw. zum Vergleich der Qualität zweier Bilder miteinander zu entwickeln. Deshalb finden sich in [Kir06] zur Bewertung der Bildqualität nicht nur PSNR- sondern auch SSIM-Messungen *(Structural Similarity)* sowie ein Vergleich von PSNR und SSIM. Während beim PSNR nur die Helligkeits- bzw. Farbwerte der Bildpunkte aus zwei Bildern miteinander verglichen werden, berücksichtigt SSIM darüber hinaus auch Ähnlichkeiten von dargestellten Strukturen. Geringe Unterschiede bei Helligkeits- bzw. Farbwerten, welche zu deutlich schlechteren PSNR-Ergebnissen führen, haben nur eine geringe Auswirkung auf die berechneten SSIM-Werte. Eine Einführung zu SSIM gibt [WBSS04], weitere Informationen zu SSIM sind auf [WWW3] zu finden.

4.1 Bündelverluste

Bündelverluste stellen bei der Videoübertragung eine schwerwiegende Form von möglichen Verlusten dar, da hierbei über einen gewissen Zeitraum überhaupt keine Daten beim Empfänger ankommen. Nach einem Bündelverlust ist in der Regel ein kompletter Neuaufbau der Bilder notwenig, der nur durch intrakodierte Makroblöcke möglich ist. Es werden die Ergebnisse der Experimente für Bündelverluste mit 10% Paketverlusten, bezogen auf die Gesamtzahl der Pakete, vorgestellt. Es wurden jeweils ab dem 1. Paket, das zu Bild 55 gehört, so viele Pakete in einem Stück verworfen, dass durch den Verlust des nächsten Pakets die 10%-Marke überschritten worden wäre. Somit waren jeweils ab Bild 55 ca. 38 Bilder von vollständigen Datenverlusten betroffen. Bei 30 Bildern pro Sekunde entspricht dies einem Ausfall der Videoübertragung von mehr als einer Sekunde.
Tabelle 2 enthält in der zweiten Spalte die Datenverluste, welche in den meisten Fällen sehr dicht an 10% liegen. Die dritte Spalte enthält die maximalen PSNR-Differenzwerte zwischen dem dekodierten Video mit und dem Video ohne Verluste. In der nächsten Spalte ist aufgeführt, nach wie vielen auf den Verlust folgenden und korrekt übertragenen Bildern der

PSNR-Differenzwert erstmals auf unter 0,5 dB fällt. Die rechte Spalte enthält die Differenzwerte der mittleren PSNR-Werte zwischen den Videos mit und ohne Verlust.
Die maximalen Δ-PSNR-Werte liegen in einem kleinen Intervall, auf das die einzelnen Kodierkonfigurationen so gut wie keinen Einfluss haben. Das liegt daran, dass die maximalen Δ-PSNR-Werte jeweils zu einem vollständig verlorenen Bild gehören und die Kodierkonfigurationen erst nach dem letzten vollständig verlorenen Bild ihre jeweilige Wirkung entfalten können. Es ist zu beachten, dass die ermittelten maximalen Δ-PSNR-Werte nicht unbedingt zu den letzten von einem direkten Verlust betroffenen Bildern gehören müssen. Da bei mehreren vollständigen Bildverlusten ab Bild 55 das letzte Bild vor den Verlusten einfach stehen bleibt und bei den Δ-PSNR-Berechnungen immer dieses stehende Bild 54 mit den Bildern des fehlerfreien Videos verglichen wird, sind die maximalen Δ-PSNR-Werte vor allem Indikatoren für die Bewegungsanteile der Videos an den von Bündelverlusten betroffenen Stellen.

Kodierkonfiguration	Datenverluste [%]	max. Δ-PSNR [dB]	Anzahl Bilder bis Δ-PSNR <0,5 dB	Δ-A-PSNR [dB]
1. IP+	9,45	27,96	-	21,11
2. IP(9)	9,50	28,83	82	5,22
3. IP+ & I-MB-Line	10,00	27,79	176	8,47
4. IP+ & I-MB 11	9,98	27,82	172	8,54
5. IP+ & FMO1	9,42	27,94	-	19,85
6. IP+ & FMO1 & I-MB-Line	10,36	27,80	171	8,99
7. IP+ & FMO1 & I-MB 11	10,54	27,86	172	8,79
8. IP+ & FMO1 & I-MB 33	9,60	27,64	64	4,17

Tabelle 2: Auswirkungen von 10% Bündelverlusten auf foreman

An der Anzahl der Bilder, die es dauert, bis der Δ-PSNR-Wert unter 0,5 dB fällt, lässt sich erkennen, dass bei den Kodierkonfigurationen ohne zusätzliche Intrakodierung, also 1 und 5, keine Erholung stattfindet. Dieses ist auch in Abbildung 2 gut zu erkennen. Die schwache Erholung, welche ungefähr ab Bild 300 einsetzt, ist zurückzuführen auf vom Kodierer selbst zusätzlich intrakodierte Makroblöcke nach einem Kameraschwenk.

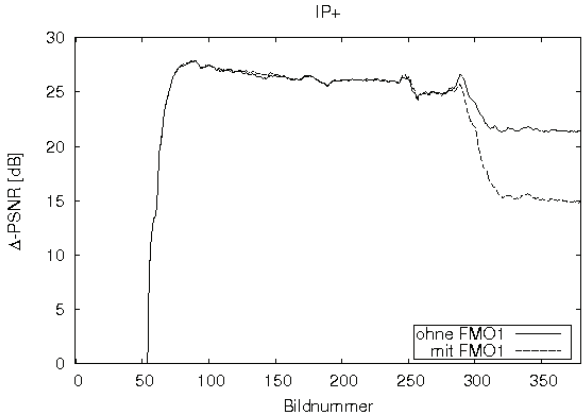


Abbildung 2: Δ-PSNR-Verläufe bei IP+-Kodierung mit und ohne FMO1

Die Erholung von den Verlusten bei den anderen Kodierkonfigurationen ist vor allem abhängig von der Anzahl und weniger von der Anordnung der zusätzlich erzwungenen I-

Makroblöcke. Dieser Umstand ist in Abbildung 3 für die Kodierkonfigurationen mit 11 zusätzlichen I-MB pro P-Bild gut zu erkennen, es gibt nur sehr kleine Unterschiede bei den Verläufen der Graphen. Bei der Kodierkonfiguration mit 33 zusätzlich erzwungenen I-Makroblöcken pro Bild findet die Erholung am schnellsten statt. Abbildung 4 links zeigt diese Erholung für FMO1 & I-MB 33.
Für die IP(9)-Kodierung zeigt sich in Abbildung 4 rechts nach den Bündelverlusten eine schnellere Erholung als bei den anderen Kodierkonfigurationen mit 11,11% zusätzlich erzwungener Intrakodierung. Nach dem letzten von einem direkten Verlust betroffenen Bild ist deutlich zu erkennen, dass die folgenden I-Bilder stets fehlerfrei dekodiert werden und sich die Fehler in den P-Bildern über diese I-Bilder hinweg fortpflanzen. Eine Kodierung mit IDR-Bildern anstelle der I-Bilder wäre hier deutlich effektiver gewesen, da sich Fehler in P-Bildern nicht über ein IDR-Bild hinweg fortpflanzen können und es spätestens 9 Bilder nach dem letzten von einem direkten Verlust betroffenen Bild, nämlich ab dem nächsten IDR-Bild, eine vollständige Erholung gegeben hätte.

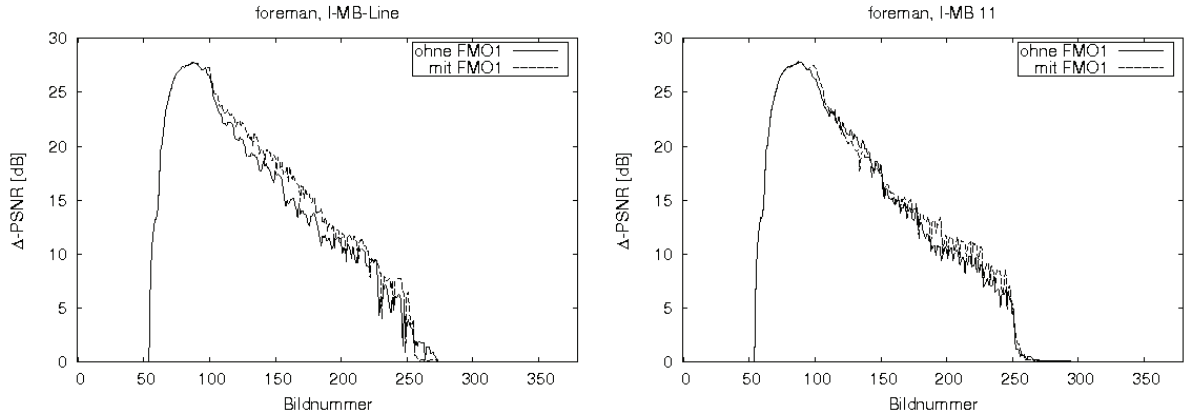


Abbildung 3: Δ-PSNR-Verläufe für foreman mit und ohne FMO1

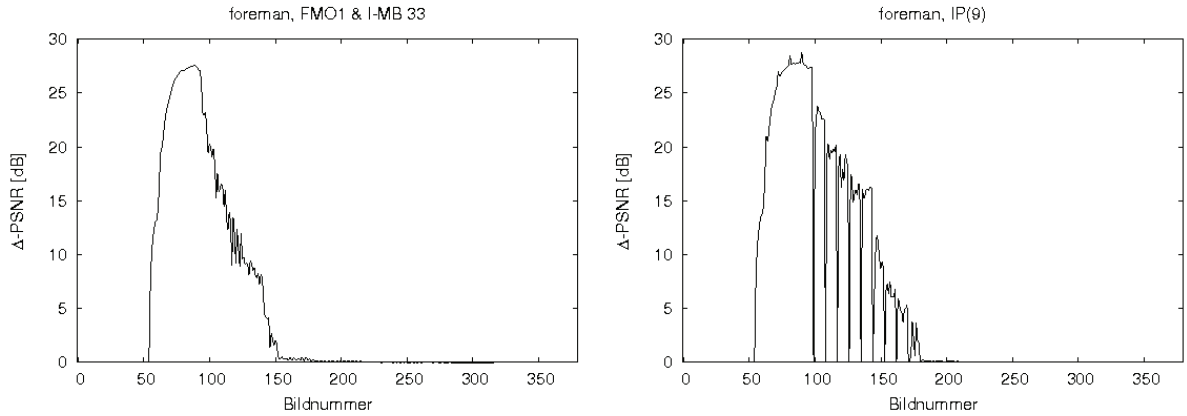


Abbildung 4: Δ-PSNR-Verläufe für foreman mit I-MB 33 (links) und mit IP(9) (rechts)

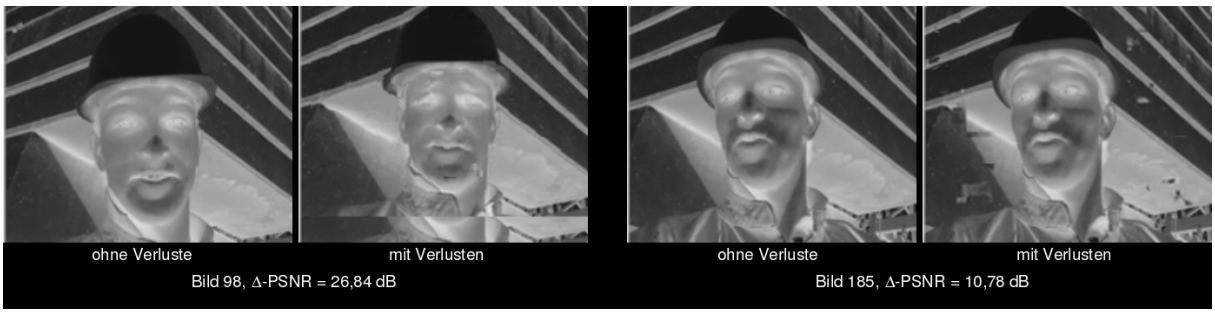


Abbildung 5: Auswirkungen von Bündelverlusten auf foreman

In Abbildung 5 sind zwei Bilder nach den Bündelverlusten dargestellt. Um die Auswirkungen der Verluste zu verdeutlichen, werden direkt neben den Bildern aus den verlustbehafteten Videos jeweils die entsprechenden Bilder aus dem verlustfreien Video gezeigt. Die Kodierkonfiguration ist IP+ & I-MB-Line. Es wird das erste Bild nach dem Bündelverlust gezeigt, das wieder vollständig übertragen wurde, sowie das Bild nach der Hälfte der Anzahl an Bildern bis der Δ-PSNR-Wert auf unter 0,5 dB gefallen ist. Der zu jedem Paar aus fehlerfreiem und fehlerbehaftetem Bild berechnete Δ-PSNR-Wert ist direkt unter den jeweiligen Bildern zu finden. Bei dem Bild direkt nach den Verlusten ist die intrakodierte unterste Bildzeile zu erkennen. Bei dem Bild ganz rechts in der Abbildung sind deutlich weniger Bildfehler im Vergleich mit dem Bild aus dem Video ohne Verluste zu erkennen als bei dem Bild direkt nach den Verlusten.

4.2 Regelmäßige Streuverluste

In diesem Abschnitt werden die Auswirkungen von Streuverlusten mit regelmäßigen Abständen vorgestellt. Es werden die Ergebnisse für Experimente mit Paketverlusten von 1% und 10% dargestellt. Dabei wurden jeweils ab Paket 11 so viele Pakete in regelmäßigen Abständen verworfen, dass das nächste verworfene Paket die jeweilige Prozentmarke überschritten hätte.

Die Auswirkungen der Streuverluste lassen sich grob in zwei Kategorien einteilen: Wenn die Anzahl der Streuverluste niedrig ist und sie relativ weit auseinander liegen, können sich die Videos zwischen den Verlusten von den Fehlern erholen. Die Verluste können dann als Folge einander nicht oder nur wenig beeinflussender Einzelverluste betrachtet werden. Wenn jedoch die Anzahl der Streuverluste höher ist, so dass die Abstände zwischen den von Verlusten betroffenen Bildern geringer sind als die Anzahl an Bildern, die es braucht, damit eine Erholung stattfinden kann, kommt es zu einer Verstärkung der Auswirkungen und die Verluste können nicht mehr wie Einzelverluste betrachtet werden. Abbildung 6 verdeutlicht diesen Umstand für die I-MB-Line Kodierkonfiguration und Paketverlusten von 1% bzw. 10%.

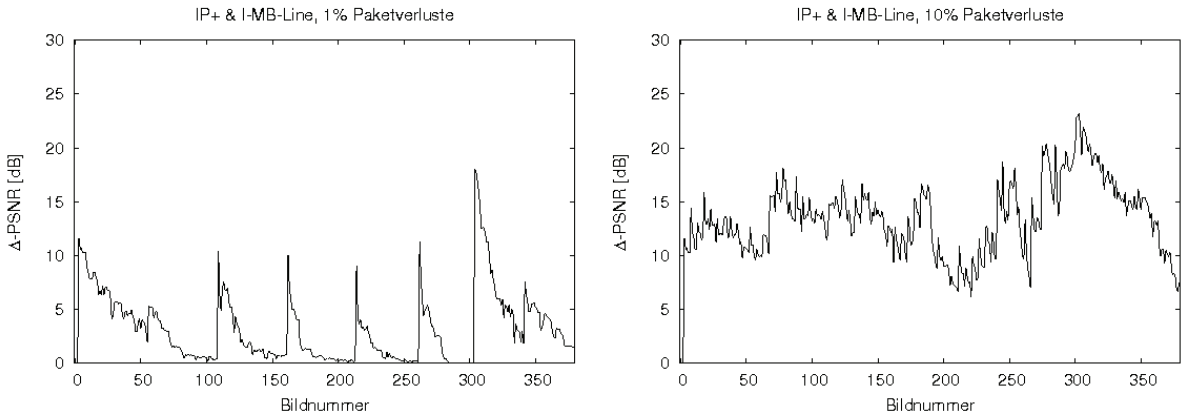


Abbildung 6: Δ-PSNR-Verläufe für regelmäßige Streuverluste mit 1% und 10% Paketverlusten

Die Auswirkungen von 1% regelmäßigen Streuverlusten fallen bei vielen Kodierkonfigurationen mit zusätzlicher Intrakodierung in die erste Kategorie. Bei mehr als 1% Paketverluste fallen, bis auf einzelne Ausnahmen, alle Kodierkonfigurationen in die zweite Kategorie.

Eine Betrachtung von maximalen Δ-PSNR-Werten oder der Anzahl der Bilder, bis die Δ-PSNR-Werte unter 0,5 dB fallen, machen bei Streuverlusten weniger Sinn. Deshalb werden in Tabelle 3 nur die Δ-A-PSNR-Werte für Paketverluste von 1% und 10% dargestellt. Für die Kodierkonfigurationen 1 und 5 haben die Δ-A-PSNR-Werte die schlechtesten Werte. Da so gut wie keine Erholung stattfindet summieren sich die Fehler nach und nach auf. Dieses ist besonders deutlich bei Paketverlusten von 1% zu erkennen. Da hier bei den anderen

Kodierkonfigurationen meistens eine vollständige Erholung zwischen den Verlusten stattfindet, sind die Unterschiede zu IP+ und FMO1 besonders groß.
Die Unterschiede zwischen den Kodierkonfigurationen mit je 11 I-Makroblöcken pro Bild ohne FMO1 und den entsprechenden Kodierkonfigurationen mit FMO1 sind gut zu erkennen. FMO1 sorgt dafür, dass maximal ein halbes Bild pro Paketverlust verloren gehen kann. Deshalb können sich die Videos von diesen Fehlern schneller erholen. Es ist anzunehmen, dass durch bessere Methoden zur Fehlerverdeckung, wie sie z.B. in [Suc01] untersucht wurden, in Verbindung mit FMO1 deutlich bessere Ergebnisse erzielt werden können. Derartige Methoden stehen in jm11.0 nicht zur Verfügung.

Kodierkonfiguration	Datenverluste [%]	Δ-A-PSNR [dB]	Datenverluste [%]	Δ-A-PSNR [dB]
1. IP+	1,09	12,92	11,30	20,54
2. IP(9)	0,86	3,66	9,65	14,72
3. IP+ & I-MB-Line	1,16	3,20	11,07	13,37
4. IP+ & I-MB 11	1,05	4,00	10,61	13,50
5. IP+ & FMO1	0,75	9,62	9,97	17,03
6. IP+ & FMO1 & I-MB-Line	0,87	2,16	9,49	10,34
7. IP+ & FMO1 & I-MB 11	1,02	2,47	9,13	10,69
8. IP+ & FMO1 & I-MB 33	0,98	0,94	10,17	6,62

Tabelle 3: Auswirkungen von 1% und 10% regelmäßigen Streuverlusten auf foreman

5 Zusammenfassung

Es wurde festgestellt, dass sich Videos ohne erzwungene Intrakodierung von Bildfehlern nach Verlusten nicht wieder erholen, es sei denn es gibt starke Veränderungen in der Bildszene, die zu einer vermehrten Kodierung von I-Makroblöcken führen. Die zusätzliche, erzwungene Kodierung von I-Makroblöcken in allen Bildern eines Videos erwies sich als effektive Maßnahme zur Verbesserung der Fehlerrobustheit eines Videos. Je mehr Makroblöcke pro Bild intrakodiert wurden, desto schneller erholte sich das Video nach Paketverlusten von Bildfehlern. Die Anordnung der I-Makroblöcke, die entweder zeilenweise oder zufällig platziert waren, hatte dabei keinen eindeutigen Einfluss auf die Ergebnisse.
Durch die gleichmäßige Verteilung der intrakodierten Makroblöcke auf alle Bilder vergrößerten sich zwar die Bildrahmen- und Dateilängen, die jeweiligen Charakteristiken der Bildrahmenlängenverteilungen und somit der Lastverteilungen wurden aber kaum beeinflusst.
Da die Kodierung mit regelmäßigen I-Bildern zu deutlichen Lastspitzen bei den I-Bildern führt, ist die Verwendung von gleichmäßig für alle Bilder zusätzlich kodierten I-Makroblöcken der Kodierung von regelmäßigen I-Bildern vorzuziehen. Wenn dennoch regelmäßige I-Bilder verwendet werden sollen, so sind IDR-Bilder den I-Bildern vorzuziehen, da sich Fehler in P-Bildern nicht über IDR-Bilder hinweg fortpflanzen können.
Es existieren in H.264/AVC Techniken zur flexiblen Aufteilung der Makroblöcke eines Bildes in mehrere, voneinander unabhängig kodierte Einheiten, die FMO genannt werden. Die durch FMO1 erzwungene Aufteilung der Bilder auf mindestens zwei Pakete verbessert die Fehlerrobustheit beim Auftreten von Streuverlusten, nicht aber beim Auftreten von Bündelverlusten.
Da in jm11.0 nur ein Teil der Fehlerrobustheitstechniken zur Verfügung stand sind weitere Untersuchungen notwendig, um ein besseres Bild der Möglichkeiten von H.264/AVC zu erhalten. Insbesondere durch den Einsatz von Kodiertechniken mit redundanten Bildern sind weitere Verbesserungen der Fehlerrobustheit von kodierten Videoströmen zu erwarten.

Darüber hinaus sollte auch das Zusammenspiel von Fehlerrobustheitstechniken, die beim Kodieren eines Videostroms beim Sender eingesetzt werden, mit Techniken zur Fehlerverdeckung, welche nach bzw. während der Dekodierung beim Empfänger eingesetzt werden, genauer untersucht werden.

Literatur

[HKS+01] Heidtmann, K., J. Kerse, T. Suchanek, B. Wolfinger und M. Zaddach: *„Fehlertolerante Videokommunikation über verlustbehaftete Paketvermittlungsnetze“,* In: Kommunikation in Verteilten Systemen (KiVS 2001) Hamburg, Springer, 2001

[ITU05] *„Advanced video coding for generic audiovisual services“,* Recommendation H.264, International Telecommunication Union, Telecommunication Standardization Sector (ITU-T), 2005

[Kir06] Kiritz, Michael: *„Untersuchung von Fehlerrobustheitstechniken für H.264/AVC-kodierte Videoströme“* , Diplomarbeit, Universität Hamburg, 2006

[Koh00] Kohlhaas, Cecil: *„Untersuchung der von Videocodierern erzeugten Verkehrslasten und der durch das Verhalten von Kommunikationssystemen beeinflussten Bildqualität“,* Diplomarbeit, Universität Hamburg, 2000

[Nor05] Norgall, Jens: *„Lastanalyse zur Modellierung H.264/AVC-kodierter Videoströme“*, MMBnet 2005, s. Wolfinger B.E., Heidtmannn K. (Hrg.), 2005, pp. 111-118

[Nor06] Norgall, Jens: *„Verkehrslast und Fehlerverhalten H.264/AVC-kodierter Videoströme in paketbasierten Kommunikationsnetzen“,* Diplomarbeit, Universität Hamburg, 2006

[Ric03] Richardson, Ian E.G.: *„H.264 and MPEG-4 Video Compression“*, John Wiley & Sons Ltd., 2003

[Suc01] Suchanek, Tim: *„Untersuchung der Bildqualität übertragener Videos in Abhängigkeit von Übertragungsverlusten und der Qualitätsverbesserung mit Hilfe von Fehlertoleranzverfahren im Empfängersystem“,* Diplomarbeit, Universität Hamburg, 2001

[WBSS04] Wang, Zhou, Alan Conrad Bovik, Hamid Rahim Sheikh and Eero P. Simoncelli: *„Image Quality Assessment: From Error Visibility to Structural Similarity“,* IEEE Transactions on Image Processing, Vol.13 No.4, April 2004

[Wen03] Wenger, Stephan: *„H.264/AVC over IP“*, IEEE Transactions on Circuits and Systems for Video Technology, Vol. 13 No. 7, July 2003

[WSBL03] Wiegand, Thomas, Gary J. Sullivan, Gisle Bjontegaard and Ajay Luthra: *„Overview of the H.264/AVC Video Coding Standard“*, IEEE Transactions on Circuits and Systems for Video Technology Vol. 13 No. 7, July 2003

[WWW1] http://iphome.hhi.de/suehring/tml/index.htm, Quellcode der H.264/AVC Referenzsoftware des JVT, aktuelle Version: jm12.2, 07.05.2007

[WWW2] http://trace.eas.asu.edu/yuv/qcif.html, Original QCIF-Videos, 07.05.2007

[WWW3] http://www.cns.nyu.edu/~lcv/ssim/, The SSIM Index for Image Quality Assessment, 07.05.2007

[Zad01] Zaddach M.: *„Modellierung, Charakterisierung und Transformation von Videoverkehrslasten“*, Dissertation, Fachbereich Informatik, Universität Hamburg, erschienen im Shaker-Verlag, Aachen 2001

[ZH01] Zaddach M. and K. Heidtmann: *„Measurement and Traffic Characterization of H.26x-coded Video Streams “*, MMB '01, Aachen, September 2001, in: Haverkort B.R., (Hrsg.), 11th GI/ITG Conference on Measuring, Modelling and Evaluation of Computer and Communication Systems, VDE-Verlag, Berlin 2001

Liste der Beitragenden