Verteiltes Messen der Dienstgüte und Netzwerk-Performance in IP-Netzen

Von der Fakultät für Elektrotechnik und Informatik der Universität Hannover

zur Erlangung des akademischen Grades

Doktor-Ingenieur

genehmigte

Dissertation

von

Dipl.-Ing. Eduard Siemens
geboren am 05.02.1969 in Uljanowskoje, Kasachstan

2005

1. Referent	Prof. Dr.-Ing. Christian Grimm
2. Referent	Prof. Dr.-Ing. Klaus Jobmann
Tag der Promotion	04.08.2005

Vorwort

Die vorliegende Arbeit entstand während meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Lehrgebiet Rechnernetze der Universität Hannover.

Mein ganz besonderer Dank geht an Herrn Prof. Grimm, nicht nur für die Ermutigung, das Thema als Dissertation zu bearbeiten. Vielmehr danke ich ihm für das Vertrauen, das er mir entgegengebracht hat, indem er mir ermöglicht hat, Probleme eigenständig zu analysieren und Lösungswege zu finden.

Ein herzlicher Dank gilt Herrn Prof. Jobmann für die Übernahme des Korreferats sowie für die hilfreichen Diskussionen zu den in der Arbeit behandelten Themen.

Herrn Dr. Hüsemann danke ich herzlich für die Unterstützung bei der Vorbereitung und Durchführung der Messungen im produktiven IP-Netz der Universität Hannover.

Meine Kolleginnen Anne-Katrin Ittmann sowie Astrid Tessmer haben ihre kritischen Anmerkungen zur Gestaltung der Arbeit und zu den Formulierungen und der Rechtschreibung gegeben. Für ihren Beitrag danke ich sehr.

Diese Arbeit wäre ohne Unterstützung meiner Familie nicht machbar gewesen. Meine Frau Tatjana und die Kinder Markus, Thomas, Anna-Liese und Ben Philipp haben mir eine angenehme Umgebung für das Schreiben dieser Arbeit geschaffen. Mein bester Dank dafür geht an sie. Ein Dank geht auch an meine Eltern Frieda und Gerhard Siemens, die mich trotz teilweise schwieriger Umstände ermutigt und es mir ermöglicht haben, die akademische Laufbahn einzuschlagen.

Hannover, den 04.08.2005

Eduard Siemens

Kurzfassung

Mit zunehmender Verbreitung des Internets werden immer mehr Anwendungen und Dienste über das Internet angeboten. Mit dem stetigen Wachstum der Vielfalt dieser Anwendungen steigt auch die Bedeutung der Durchführung von Messungen in IP-Netzen. Mit Hilfe von Messungen können Engpässe im Datennetz aufgespürt werden. Aus den Messergebnissen können anschließend Maßnahmen für die Optimierung der Verteilung von Netzressourcen abgeleitet werden.

In der vorliegenden Arbeit werden Aspekte der Messung der Netzwerk-Performance sowie der Dienstgüte in IP-Netzen beleuchtet. Es wird eine Auswahl von Parametern getroffen, welche die Qualität der im Netz implementierten Anwendungen beeinflussen. Dabei stehen Anwendungen audiovisueller Kommunikation mit Echtzeit-Anforderungen sowie das Grid-Computing im Vordergrund der Betrachtungen.

Mit der getroffenen Auswahl betrachteter Parameter der Dienstgüte sowie der Netzwerk-Performance wird ein geeignetes Szenario für die Durchführung von Messungen erarbeitet. Es wird dabei gezeigt, dass eine Durchführung von aktiven Einweg-Messungen besser als andere Verfahren für die Bestimmung der Dienstgüte geeignet ist. Des Weiteren wird festgestellt, dass die Behandlung der Datenströme im Netz von deren Verkehrscharakteristiken abhängt. Zu diesen Charakteristiken gehören unter anderem die Verteilung der Zwischenankunftszeiten der Datenpakete, die Verteilung der Paketlängen sowie das verwendete Transportprotokoll.

Nachdem die Methode für das Messen im Netz festgelegt worden ist, werden typische Szenarien vorgestellt, in welchen Nutzer von Diensten sowie Netz- und Dienstanbieter die Parameter des Netzes bestimmen können. Aus den dargestellten Szenarien geht die Notwendigkeit der Realisierung verteilter Messungen zwischen mehreren Punkten im Netz hervor. Eine Möglichkeit zur Realisierung eines verteilten Messsystems wird aufgezeigt. Anschließend werden Maßnahmen zur Reduzierung der Messfehler bei aktiven Einweg-Messungen vorgestellt.

Mit einer prototypischen Implementierung des Softwaresystems *LTest* wird ein Messsystem vorgestellt, mit dem verteilte aktive Messungen in IP-Netzen durchgeführt werden können. Mit diesem System wird in einer Labor-Umgebung die Auswirkung erarbeiteter und implementierter Konzepte vorgestellt.

LTest ist für den Einsatz auf Standard-PC-Hardware sowie Unix-Betriebssystemen wie Linux und Solaris implementiert worden. Auf Veränderungen im Betriebssystemkern ist dabei verzichtet worden. Damit kann das Messsystem für Test-Zwecke verwendet werden. Mit Hilfe von Messungen werden die Grenzen der Güte einer Realisierung eines

hochpräzisen Messsystems auf einem nicht echtzeitfähigen Betriebssystem aufgezeigt worden.

Anschließend werden am Beispiel einiger Messungen in einem produktiven Netz die Anwendung des Systems sowie die Interpretation der Ergebnisse und deren Abhängigkeit vom verwendeten Verkehrsprofil der Mess-Datenströme gezeigt.

Abschließend wird eine Zusammenfassung sowie ein Ausblick auf die zukünftig durchzuführenden Untersuchungen und Entwicklungen gegeben.

Schlagwörter: Netzwerk-Performance, Dienstgüte, Messung, Messen, Einweg-Messung

Abstract

With the growing popularity of the Internet, more and more applications and services are being implemented and offered over the network. Many of them need high performance networks because of the real-time character of these applications and services. The conventional Internet lacks the means to prevent service starvation for such applications. Bottlenecks can only be discovered by means of network monitoring and performance investigations. This demands new techniques for network measurements.

In this thesis, aspects of measurements for estimating quality of service and the IP network performance are discussed. A set of performance related parameters which have impact on the quality of applications in the network is chosen. Hereby the main focus lies on demands for real-time multimedia as well as for Grid applications.

With these demands, a suitable measurement scenario to assess the quality of service, is developed. This thesis states that an active one-way measurement scenario is suitable for this purpose. Secondly, it is stated, that the behavior of data flows in network nodes is dependent on the characteristics of these flows. These characteristics include the distribution of packet sizes as well as interpacket time distribution and used transport protocol.

After the selection of a suitable measurement method, typical scenarios for measurements are outlined. In these, users as well as network and service providers can assess the network performance and the quality of service. The demands of users and providers is analyzed and this reveals the need of a distributed measurement approach. Additionally, a way to implement such measurements is worked out. Certain procedures to reduce measurement inaccuracy are discussed.

Furthermore, a software tool is developed and implemented for a distributed active one-way measurement approach. With this tool, the impact of chosen measurement parameters and procedures for reducing inaccuracy on the results are being shown by running the tool in a testbed.

The implemented tool called *LTest* is developed to run on standard hardware and on a Unix-based operating system without changes in the kernel. With *LTest*, high performance measurements are taken in a laboratory. Bounds of accuracy on a non real-time OS are tested and outlined.

Finally, the application of the system to the real world network is shown by examples of certain measurements at the edge between a corporate network and the the Internet.

To conclude, a discussion of measurement results takes place, showing the dependence of the results on the chosen scenario.

Key words: network, measurement, performance, QoS, service quality, Quality of Service

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
ASP	Application Service Provider
ATM	Asynchronous Transfer Mode
BSD	Berkeley Software Distribution
BTC	Bulk Transfer Capacity
CAIDA	Cooperative Association for Internet Data Analysis
CBR	Constant Bit Rate
CCDF	Complementary Cumulative Distribution Function
CDF	Cumulative Distribution Function
CGI	Common Gateway Interface
CORBA	Common Request Broker Architecture
CPU	Central Processing Unit
CSV	Comma Separated Value
DCF77	Langwellen-Sender mit der Referenz-Zeit, welche auf der Trägerfrequenz 77,5 kHz ausgestrahlt wird
DFN	Verein zur Förderung eines Deutschen Forschungsnetzes e.V.
DiffServ	Differentiated Services
DNS	Domain Name System
DOM	Document Object Model
DSCP	DiffServ Codepoint

DTD	Document Type Definition
EGEE	Enabling Grids for E-science in Europe
FIFO	First In First Out
FTP	File Transfer Protocol
G-WiN	Gigabit-Wissenschaftsnetz
G.261	Ein Standard der ITU-T zur Codierung von Videosequenzen
G.263	Ein Standard der ITU-T zur Codierung von Videosequenzen
GCC	Gnu Compiler Collection
GPS	Global Positioning System
GUI	Graphical User Interface
H.225	Ein Signalisierungsprotokoll der H.323-Protokollfamilie
H.245	Ein Signalisierungsprotokoll der H.323-Protokollfamilie
H.323	Eine Protokollfamilie der ITU-T für audiovisuelle Kommunikation
HLRN	Hochleistungsrechner-Verbund Nord
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IEEE	Institute for Electrical and Electronic Engineers
IETF	Internet Engineering Task Force
IntServ	Integrated Services
IP	Internet Prorocol
IPDV	IP Packet Delay Variation
IPPM	IP Performance Metrics
ISDN	Integrated Services Digital Network
ISP	Internet Service Provider

ITG	Informationstechnische Gesellschaft im VDE
ITSP	IP Telephony Service Provider
ITU	International Communication Unit
ITU-T	ITU Telecommunication Standardization Sector
KDE	K Desktop Environment
LAN	Local Area Network
MGCP	Media Gateway Control Protocol
MICO	Rekursive Abkürzung für „MICO is CORBA“ – eine freie CORBA-Implementierung
MPEG	Moving Pictures Experts Group
MSB	Most Significant Byte
NLANR	National Laboratory for Applied Network Research, USA
NNTP	Network News Transport Protocol
NTP	Network Time Protocol
NTPD	NTP-Daemon
ODBC	Open Database Connector
OMG	Object Management Group
OSI	Open Systems Interconnect
OVSt	Ortsvermittlungsstelle
OWAMP	One-way Active Measurement Protocol
OWD	One-way Delay
PC	Personal Computer
PDF	Probability Distribution Function
PDU	Protocol Data Unit
QoS	Quality of Service

RAM	Random Access Memory
RFC	Request for Comments
RIPE	Réseaux IP Européens
RIPE NCC	Réseaux IP Européens Network Coordination Centre
RSVP	Resource Reservation Protocol
RTD	Round-trip Delay
RTP	Real-time Transport Protocol
RTT	Round-trip Time
SAX	Serial Access Parser for XML
SDH	Synchronous Digital Hierarchy
SI	Système International d'Unités
SIP	Session Initiation Protocol
SLA	Service Level Agreement
SOAP	Simple Object Access Protocol
SPARC	Scalable Processor Architecture
SQL	Structured Query Language
SSL	Secure Socket Layer
SVG	Scalable Vector Graphics
TCP	Transport Control Protocol
TFTP	Trivial File Transfer Protocol
TLS	Transport Layer Security
TTL	Time To Live
UDP	User Datagram Protocol
URL	Unified Resource Locator

UTC	Coordinated Universal Time
VoIP	Voice over IP
W3C	World Wide Web Consortium
WAN	Wide Area Network
WiN	Wissenschaftsnetz
WWW	World Wide Web
XML	Extensible Markup Language

Kapitel 1

Einleitung

„Das Messen ist die einzige legitime Methode zum Gewinnen naturwissenschaftlicher Erkenntnisse“

Ludwig Merz [Mer77, S. 14]

1.1 Motivation

Information ist eine fundamentale Grundgröße vieler Wissenschaften. Jedes intelligente System ist im Besitz von Information [Git02, Kap. 4]. Für Unternehmen, Organisationen und Institutionen sowie für ganze Branchen ist es essentiell wichtig, Informationen zu besitzen und in der Lage zu sein, diese zwischen unterschiedlichen Orten auszutauschen. Für den Informationsaustausch werden derzeit überwiegend digitale Datennetze verwendet. Dabei wird die Information an einer Quelle digitalisiert und als eine geordnete Menge binär codierter Daten zum Ziel übertragen. Am Ziel werden die Daten empfangen, ausgewertet und weiterverarbeitet.

Die rapide Entwicklung digitaler rechnergestützter Datenverarbeitung der letzten 20 Jahre sorgte für eine ebenso schnelle Weiterentwicklung der Datennetze. Die Steigerung des übertragenen Datenvolumens kann am Beispiel der Nutzungsstatistik des Vereins zur Förderung eines Deutschen Forschungsnetzes e.V. (DFN) mitverfolgt werden. Hier wurden aus dem Wissenschaftsnetz (WiN) im Januar 1996 rund 2.600 GByte in das restliche Internet übertragen [DFN96]. Acht Jahre später, im Januar 2004, waren es schon ca. 1.300 TByte [DFN04]. Dies entspricht einer jährlichen Steigerung des übertragenen Datenvolumens um einen Faktor größer als 2.

Der Anstieg des übertragenen Datenvolumens treibt wiederum die Weiterentwicklung sowie den Ausbau der Datennetze voran. Mit dem Ausbau der Netze werden neue Anwendungen, welche einen intensiven Datenaustausch erfordern, im Netz etabliert. Als Beispiele dafür können das HLRN-Projekt [HLR03], der Videokonferenz-Dienst des DFN [Mai01, S.13] sowie das EGEE-Projekt [EGEE04] dienen.

Der digitale Datenaustausch findet nach bestimmten Regeln statt. Diese werden in der Fachwelt als *Kommunikationsprotokolle* oder einfach *Protokolle (engl. Protocol)*

bezeichnet. Bei der großen Vielfalt verwendeter Kommunikationsprotokolle spielt das *Internet Protocol (IP)* [Pos81a] wegen seiner weiten Verbreitung eine herausragende Rolle. Netze, in welchen ein Datenaustausch nach dem IP-Protokoll abläuft, werden als IP-Netze bezeichnet. Das Internet ist dabei das weltweit größte IP-Netz.

Folgende Faktoren beeinflussen die aktuellen Entwicklungen der IP-Netze maßgeblich:

- Die Dienstgüte im Internet wurde ursprünglich nach dem so genannten *Best-Effort*-Prinzip realisiert. Dies bedeutet, dass die Netzressourcen allen Teilnehmern bzw. Anwendungen im Netz in gleichem Maße zur Verfügung stehen. Bei diesem Verfahren wird nicht sichergestellt, dass alle Anforderungen im Netz erfüllt werden. Die Vermittlung von Datenpaketen in modernen IP-Netzen läuft immer noch überwiegend nach dem Best-Effort-Prinzip ab.

  Dennoch werden derzeit in IP-Netzen Anwendungen realisiert, welche bestimmte Mindestanforderungen hinsichtlich der Performance des Netzes stellen. Dazu gehören z.B. IP-Telefonie, Videokonferenz-Dienste, Datenbank- sowie Grid-Anwendungen.

  Eine adäquate Qualität der Anwendungen kann dabei nur durch sorgfältige Planung und Dimensionierung der Netzressourcen sichergestellt werden. Dies wiederum setzt sehr gute Kenntnisse der Eigenschaften von Anwendungen sowie ebenso gute Kenntnisse der Auslastung des Netzes seitens der Netzbetreiber voraus.

- Im Gegensatz zu den so genannten *leitungsvermittelten* Netzen [Sie99a, Cla99] sind im IP-Protokoll keine Schnittstellen für ein direktes Abfragen bestimmter Leistungsparameter des Netzes – so genannte Management-Schnittstellen – vorgesehen. Kenntnis über das Netz bzw. über die Behandlung der Datenströme im Netz kann nur auf zwei Wegen gewonnen werden:

  1. durch die Beobachtung und Analyse der Datenströme an bestimmten Stellen im Netz. Solche Beobachtung des Netzes wird als passives Messen bezeichnet [Zse02]. Sie ist für eine Ist-Aufnahme der Netzzustände gut geeignet.
  2. durch gezieltes Injizieren von Datenpaketen – so genannten Probepaketen – in das Netz. Dabei werden die Zeitpunkte für das Aussenden und das Empfangen von Datenpaketen beobachtet und daraus die Laufzeiten der Datenpakete sowie Paketverluste ermittelt. Diese Methode wird als aktives Messen bezeichnet [Pas01]. Dabei wird in der Regel ein Datenstrom aus Probepaketen erzeugt und anschließend das Ergebnis der Messung ausgewertet. Mit Hilfe von aktiven Messungen kann die erwartete Güte einer Anwendung im Netz abgeschätzt werden.

- Mit zunehmender Kommerzialisierung der Nutzung der IP-Netze gewinnen so genannte *Service Level Agreements (SLA)* [Mar02] an Bedeutung. Diese bilden eine

Grundlage der Beziehung zwischen Dienstanbietern und Nutzern im Internet. Die SLA beinhalten unter anderem eine technische Beschreibung der Güte der angebotenen Dienste. Die Einhaltung vereinbarter Parameter kann wiederum nur mit Hilfe der oben dargestellten Methoden zum Messen im Netz überprüft werden. Dabei steht den Nutzern von Diensten im Netz in der Regel nur die zweite Methode zur Verfügung, da die erste einen Zugang zu mehreren Netzkomponenten voraussetzt.

- Das globale Internet stellt eine Zusammenschaltung einzelner IP-Netze dar. Diese sind hinsichtlich der administrativen Verantwortung unabhängig voneinander. Damit ist für die Netzbetreiber ein administrativer Zugriff auf benachbarte Netze nicht möglich. Die Güte des Netzes zwischen zwei Endpunkten kann dabei nur mittels einer Ende-zu-Ende-Messung bestimmt werden.

Die bereits im frühen Stadium der Entwicklung von IP-Netzen realisierten Verfahren und Werkzeuge für die Durchführung von Messungen (siehe Kapitel 2.7) bieten lediglich eine grobe Abschätzung der Dienstgüte. Dies war und ist für die ursprünglichen Dienste des Internets, wie z.B. Mail, News- sowie das World Wide Web [Pos82, Kan86, Fie99], häufig ausreichend. Im Vordergrund dieser Messungen steht dabei die Bestimmung einer rudimentären Konnektivität zwischen den an der Verbindung beteiligten Systemen. Paketverlustraten von mehreren Prozent waren dabei beispielsweise dank der Verwendung des *Transport Control Protocol (TCP)* [Pos81c, Jac92, Ste97a] durchaus tolerierbar.

Die in den letzten Jahren stattgefundene Weiterentwicklung der IP-Netze stellt neue Anforderungen an die verwendeten Werkzeuge für das Messen ebenso wie an die Methodik der Durchführung von Messungen. Die Veränderung der Anforderungen hat primär folgende Ursachen:

- Häufig haben die über IP realisierten Anwendungen sehr hohe Anforderungen an die Güte des Netzes. So können z.B. Schwankungen der Einweg-Verzögerung von einigen wenigen Millisekunden in einer wahrgenommenen Verringerung der Qualität des Dienstes bemerkbar werden. Häufig bringen bereits Paketverlustraten von unter 1 % die Netzdienste zum Erliegen. Dabei spielen unter anderem Paketverlust-Muster (siehe Kapitel 2.3.5) eine Rolle.

- Die Netze werden stetig ausgebaut. In IP-Netzen werden Übertragungsraten von mehreren GBit/s erreicht. Dies erfordert nicht nur eine Steigerung der Performance verwendeter Messsysteme, sondern auch die Verwendung einer zeitlichen Auflösung der Betrachtungen im Bereich von wenigen Mikrosekunden.

- Verschiedene Anwendungen haben sehr unterschiedliche Anforderungen an die Kommunikationsnetze. Eine qualitative Aussage über die Güte des Netzes kann

nur in Verbindung mit einer betrachteten Anwendung gemacht werden. Folglich sind bei der Durchführung von Messungen die Eigenschaften einzelner Anwendungen zu berücksichtigen.

1.2 Zielsetzung

Ziel dieser Arbeit ist es, einen systematischen Ansatz für die Realisierung von Messungen in IP-Netzen zu erarbeiten, welcher den Anforderungen moderner Anwendungen in Hochgeschwindigkeitsnetzen entspricht. Dabei sind Parameter festzulegen, welche die Güte von IP-Netzen repräsentativ beschreiben. Aufbauend auf den festgelegten Parametern sollen geeignete Messszenarien und Methoden zur Messung der Dienstgüte sowie der Netzwerk-Performance erstellt werden.

Für die Evaluierung des erarbeiteten Ansatzes soll ein Werkzeug für die Realisierung von Messungen entworfen und prototypisch implementiert werden. Dieses soll möglichst flexibel und universell einsetzbar sein. Dafür ist eine Wahl der Hardware- und Softwareplattform zu treffen, auf welcher das Messsystem zu realisieren ist. Diese Auswahl soll unter dem Gesichtspunkt der Verfügbarkeit der Hard- und Software getroffen werden. Das Messsystem soll für die Untersuchung der Dienstgüte und der Netzwerk-Performance in produktiven Netzen sowie in der Forschung und Entwicklung von Datennetzen einsetzbar sein. Das Messsystem soll die Anforderungen von Dienstanbietern und Netzbetreibern, aber auch von Nutzern von IP-Netzen berücksichtigen, wobei bei Nutzern grundlegende Kenntnisse von Netzen und Rechnersystemen vorausgesetzt werden.

Die Installation und Bedienung des Messsystems soll keine speziellen Programmierkenntnisse erfordern. Dafür sollen, soweit möglich, Veränderungen im Betriebssystem sowie eine Abhängigkeit von speziellen Treibern vermieden werden. Das System soll erweiterbar sein, damit es zukünftig durch Weiterentwicklung in Umgebungen zum Netzwerk-Monitoring in verteilten Szenarien einsetzbar wird.

Anschließend soll eine Fehlerabschätzung der Messungen mit dem erarbeiteten Messszenario durchgeführt werden. Ebenso sind mögliche Einsatzszenarien zu nennen sowie die Grenzen der Performance des Systems zu bestimmen.

1.3 Aufbau der Arbeit

In Kapitel 2 werden die Begriffe in Verbindung mit dem Messen der Dienstgüte erläutert. Die bereits in der Einleitung verwendeten Begriffe *Quality of Service* und *Netzwerk-Performance* werden präzisiert. Des Weiteren werden die für diese Arbeit relevanten

Protokolle und Standards genannt. Anschließend wird auf die im Umfeld der Netzmessungen wesentlichen Initiativen sowie auf bereits existierende Messtools eingegangen.

In Kapitel 3 wird eine umfassende Analyse der Anforderungen an eine Umgebung für das Messen der Dienstgüte sowie der Netzwerk-Performance gegeben. In diesem Zusammenhang wird auch die Problematik der Uhrensynchronisation behandelt. Es wird ein geeignetes Messszenario festgelegt. Des Weiteren wird eine Fehlerabschätzung im gewählten Messszenario gegeben. Ebenso werden die Anforderungen an die Realisierung diverser Quellenmodelle für die Durchführung von aktiven Messungen gegeben.

Kapitel 4 beschäftigt sich mit der Implementierung des Messtools *LTest*. Es wird die Kommunikationsstruktur des Programms vorgestellt. Ebenso werden die Schnittstellen zu anderen Programmen erläutert. Außerdem werden Maßnahmen zur Reduzierung der Messfehler vorgestellt. Abschließend wird auf die Visualisierung der Messergebnisse eingegangen.

Mit der Darstellung der Evaluierungsergebnisse von *LTest* werden in Kapitel 5 die Ergebnisse von mehreren Messungen in einer Labor-Umgebung sowie in einem produktiven Netz vorgestellt. Dabei steht die Auswirkung realisierter Konzepte auf die Güte und Performance des Messsystems im Vordergrund. Anschließend werden einige typische Messszenarien beschrieben. Die dargestellten Ergebnisse zeigen die Grenzen für die Güte eines Messsystems auf, welches auf einer nicht echtzeitfähigen Hardware und Software implementiert ist.

Zusammenfassend werden in Kapitel 6 die Vorteile der in dieser Arbeit entwickelten Ansätze für die Messung von Dienstgüte sowie der Netzwerk-Performance gegenüber bereits existierenden Verfahren genannt. Es werden zukünftige Einsatzmöglichkeiten für *LTest* beschrieben sowie ein Ausblick auf mögliche Erweiterungen der erarbeiteten Methodik der Messung gegeben.

Im Anhang der Arbeit werden begleitende Untersuchungen vorgestellt, die für den Entwurf und die Entwicklung von *LTest* zusätzlich durchgeführt wurden. Ebenso wird ein Tool zur Bestimmung statistischer Maßzahlen und Verteilungen von Zwischenankunftszeiten und Paketlängen von Datenströmen in IP-Netzen vorgestellt.

Kapitel 2

Grundlagen

2.1 Dienstgüte und Netzwerk-Performance

Der Begriff *Dienstgüte* – auch als *Quality of Service (QoS)* bezeichnet – steht in einem engen Zusammenhang mit dem Messen in Datennetzen. Die Dienstgüte kann mittels Messungen quantifiziert werden. Ebenso können Messungen die Einhaltung bestimmter Gütekriterien verifizieren. Aus den Messergebnissen können ggf. Maßnahmen zur Verbesserung der Dienstgüte bzw. der Netzwerk-Performance abgeleitet werden. Die Begriffe *Quality of Service* und *Netzwerk-Performance* werden in der Literatur teilweise diffus interpretiert, so dass hier eine Klarstellung erforderlich ist.

2.1.1 Dienstgüte und Netzwerk-Performance bei der ITU-T

In den Dokumenten der *International Telecommunication Unit (ITU)* wird der Begriff *Dienstgüte* sehr weitläufig angewendet. Der Standard X.200 der ITU-T [ITU94a] beschreibt beispielsweise QoS als einen Satz von Parametern in Bezug auf die Datenübertragung in einer der sieben OSI-Schichten [ITU94a]. Im weitesten Sinne gehören dabei zu QoS beinahe alle Aspekte der Nutzung der Netze seitens der Kunden – die Qualität der Kundenbetreuung ebenso wie die des Fehlermanagements beim Erbringer eines Dienstes. In den Standards X.641 und X.642 [ITU97b, ITU98] wird dieses Konzept weiter verfeinert. So werden nicht direkt messbare Parameter, wie z.B. Kundenzufriedenheit, Nutzbarkeit etc., von objektiven, messbaren Parametern wie Durchsatz, Paketverzögerung und Jitter, getrennt. Semantisch wird Quality of Service in den Dokumenten der ITU-T in drei Bereiche unterteilt:

1. **QoS-Maßnahmen**
   Unter QoS-Maßnahmen versteht man Maßnahmen, die getroffen werden, um die Realisierung eines Dienstes zu ermöglichen, bzw. durch welche die Einhaltung bestimmter Parameter sichergestellt wird.

2. **QoS-Charakteristiken**
   QoS-Charakteristiken sind messbare Größen, die das Verhalten der Datenströme

im Netz beschreiben. QoS-Charakteristiken werden unter anderem bei der Definition von *Service Level Agreements (SLA)* zwischen Anbietern von Diensten und deren Nutzern angewendet. Die ITU-T legt keinen festen Satz relevanter QoS-Charakteristiken fest. Diese Festlegung kann nur in Bezug auf ein konkretes Netz stattfinden. Des Weiteren hängt die Wahl der Charakteristiken von der betrachteten Anwendung und von der zu bearbeitenden Fragestellung ab. Es können auch abgeleitete Parameter, wie z.B. statistische Maßzahlen betrachteter Parameter, zur QoS-Charakterisierung verwendet werden. Die genaue Festlegung für die in dieser Arbeit relevanten Parameter findet in Kapitel 2.4 statt.

3. **QoS-Anforderungen**
   Häufig wird QoS bei der ITU-T als Satz von Anforderungen seitens der Anwendungen an das Netz definiert. Diese Anforderungen können unter anderem von regulatorischen Bestimmungen geprägt sein. Im Allgemeinen sind die Anforderungen aber eng mit der subjektiven Betrachtung der Qualität eines Dienstes verknüpft. Bei multimedialen Anwendungen wird dabei oft auf Erfahrungswerte zurückgegriffen, die in Fachkreisen unterschiedlich bewertet werden können. Speziell im Fall der Sprachübertragung über isochrone Netze gibt es Richtwerte für Übertragungsverzögerungen und Verluste von Samples, die auf Untersuchungen in den Bell Laboratories der AT&T der 50er bis Mitte 70er Jahre zurückführen. Diese sind in den ITU-T-Standards G.113 und G.114 [ITU96a, ITU96b] beschrieben. Die Übertragung von Sprachdaten in IP-Netzen ist aber keineswegs isochron. Somit ist eine direkte Anwendung der in diesen Standards festgelegten Richtwerte nicht möglich. Einige Untersuchungen der Auswirkungen von Störungen in IP-Netzen auf die Güte der Sprachanwendungen sowie der Videoübertragung sind in [Göh01] und [Hei01] beschrieben.

2.1.1.1 Unterschied zwischen Dienstgüte und Netzwerk-Performance

Die Definition der *Netzwerk-Performance* laut ITU-T-Standard E.800 [ITU94b] ist ähnlich der in Abschnitt 2, Kapitel 2.1.1, beschriebenen Bedeutung der Dienstgüte – es wird ebenfalls ein Satz von Parametern betrachtet, welcher das Netz quantitativ beschreibt. Der wesentliche Unterschied zur Dienstgüte liegt darin, dass bei der Betrachtung der Dienstgüte die Realisierung einer konkreten Anwendung im Vordergrund steht. Dieser Betrachtung wird ein Ende-zu-Ende-Szenario zugrunde gelegt. Die Parameter der Netzwerk-Performance werden dagegen von den Netzbetreibern zur Planung und Dimensionierung ihrer Netze herangezogen. Die Netzwerk-Performance wird durch die Messung bestimmter Parameter des Netzes verifiziert. Die Messungen beziehen sich dabei nicht zwangsläufig auf ein Ende-zu-Ende-Szenario, sondern auch auf bestimmte Abschnitte im Netz.

2.1.2 Dienstgüte und Netzwerk-Performance bei der IETF

Die Entwicklung von Verfahren und Protokollen in IP-Netzen wird von der *Internet Engineering Task Force (IETF)* [IETF04] koordiniert. In der vorliegenden Arbeit stehen Messungen in IP-Netzen im Vordergrund, folglich ist die Sicht der IETF bezüglich Netzmessungen hier von besonderer Bedeutung. Anders als bei der ITU-T ist der Begriff *Quality of Service* in den IETF-Dokumenten weniger formal beschrieben.

In RFC 2386 [Cra98] wird QoS als ein Satz von Anforderungsparametern definiert, welche das Netz zu erfüllen hat. Dies entspricht in etwa der Definition der QoS-Anforderungen der ITU-T (Kapitel 2.1.1). Unter QoS im IETF-Umfeld werden aber auch Verfahren verstanden, die dazu dienen, die Einhaltung geforderter Parameter der Dienstgüte sicherzustellen. Eine ausführliche Analyse dieser Verfahren sowie deren Anwendung in IP-Netzen wird in [Bur01] gegeben.

Der Begriff *Netzwerk-Performance* wird in den Dokumenten der IETF in Verbindung mit bestimmten gemessenen Parametern des Netzes verwendet. Es gibt dort aber keine formale Definition des Begriffs. Dieser wird häufig im Zusammenhang mit der Beschreibung der Leistungsfähigkeit von Netzwerk-Komponenten sowie von Verbindungen in IP-Netzen verwendet.

2.1.2.1 IP Performance Metrics Working Group

Die steigenden Anforderungen an das Netz erfordern neue Methoden zur Messung der Güte des Netzes sowie die Definition eindeutiger und aussagekräftiger Messgrößen. Diese Problematik adressiert die 1999 ins Leben gerufene *IP Performance Metrics Group (IPPM)* [IPP04] der IETF.

Damit Messergebnisse, die an verschiedenen Orten bzw. zu verschiedenen Zeiten gewonnen worden sind, miteinander verglichen werden können, müssen bei der Durchführung der Messungen zwei wesentliche Bedingungen erfüllt sein:

1. Es muss definiert sein, was die quantifizierte Größe beinhaltet. So müssen zum Beispiel bei der Angabe von Einweg-Verzögerung die Ereignisse „das Aussenden“ und „das Empfangen“ der Datenpakete klar sein [Alm99a, Kap. 3.4].
2. Es müssen alle Nebenbedingungen, welche die Messergebnisse beeinflussen, bekannt sein und bei der Beschreibung der Messung genannt werden.

Sind für eine Messgröße die genannten Bedingungen erfüllt, so wird diese Messgröße von der IPPM als Metrik bezeichnet.

Für die Klärung der Begriffe in Verbindung mit Messungen in IP-Netzen wurde von der IPPM als Erstes ein Rahmenwerk [Pax98a] für die Definition geeigneter Metriken erstellt. Dieses legt die Anforderungen an die zu definierenden Größen fest, damit diese als Metriken verwendet werden können. Außerdem werden in dem Rahmenwerk die Aspekte der Uhrensynchronisation sowie die Fehlerquellen bei der Zeitmessung behandelt. Diese beiden Themen werden in den Kapiteln 2.5.2 und 3.2.5.1 der vorliegenden Arbeit behandelt. Im Einklang mit dem Rahmenwerk wurde von der Arbeitsgruppe ein Satz performancerelevanter Metriken definiert. Die von dieser Arbeitsgruppe definierten Metriken sind in Tabelle 2.1 dargestellt. Einige davon stellen eine ganze Klasse von Metriken dar, die von einer Basis-Metrik abgeleitet sind. Diese sind in der dritten Spalte der Tabelle entsprechend gekennzeichnet. Die entsprechenden Spezifikationen sind in [Mah99, Alm99a, Alm99b, Koo02, Alm99c, Dem02, Mor04] zu finden.

Metrik bzw. Metriken	**Aspekt der Netzwerk-Performance**	**abgeleitete Metriken**
Unidirectional Connectivity	Konnektivität einer Verbindung zwischen zwei Endsystemen	ja
One-way Delay	Einweg-Verzögerung der Datenpakete zwischen zwei Endsystemen	nein
One-way Packet Loss	Paketverlust auf dem Weg zwischen zwei Endsystemen	ja
One-way Loss Pattern	Burstiness des Paketverlustes	ja
Round-trip Delay	Umlauf-Verzögerung der Datenpakete zwischen zwei Endsystemen	nein
IP Packet Delay Variation	Jitter der Laufzeiten im Netz	nein
Packet Reordering	Paketüberholung im Netz	nein

Tabelle 2.1: Metriken der IPPM

Des Weiteren beschäftigt sich die IPPM Group mit der Entwicklung eines Protokolls für aktive Einweg-Messungen – des *One-way Active Measurement Protocol (OWAMP)* [Sha04]. Dieses soll in erster Linie der Bestimmung der Konnektivität sowie der Parameter von Strecken zwischen Netzknoten [Luc01] und nicht der Bestimmung Ende-zu-Ende-Dienstgüte dienen [Var04] (siehe auch Kapitel 2.7).

2.1.3 Begriffe Dienstgüte und Netzwerk-Performance in dieser Arbeit

In den vorhergehenden Abschnitten wurde die unterschiedliche Verwendung der Begriffe *Dienstgüte (Quality of Service)* und *Netzwerk-Performance* in den Dokumenten der ITU-T sowie der IETF vorgestellt. Aus der unterschiedlichen Verwendung folgt die Notwendigkeit zur Klärung dieser Begriffe in der vorliegenden Arbeit.

Entsprechend der Zielsetzung werden hier nur quantifizierbare Parameter des Netzes bzw. einer Anwendung betrachtet, welche die wahrgenommene Qualität einer im IP-Netz realisierten Anwendung beeinflussen. Dabei werden die Phasen des Aufbaus, der Informationsübertragung sowie des Verbindungsabbaus betrachtet (siehe [Ood97, S.22]). Zu beachten ist dabei, dass mit dem Aufbau und Abbau von Verbindungen im Internet keine Reservierung von Ressourcen im Netz vorgesehen ist und sich somit diese Phasen von den Phasen in der klassischen Telekommunikation unterscheiden.

Definition 2.1 *Mit Quality of Service (QoS) wird in dieser Arbeit ein Satz von Parametern bezeichnet, welcher das Netz bzw. eine Anwendung im Netz beschreibt. Dabei müssen die verwendeten Parameter quantifizierbar sein und die wahrgenommene Qualität einer Anwendung beeinflussen.*

Definition 2.2 *Mit Netzwerk-Performance (engl. Network Performance) wird in dieser Arbeit ein Satz von quantifizierbaren Parametern bezeichnet, welcher das Netz oder Teile davon hinsichtlich der Tauglichkeit für den Transport bestimmter Anwendungen beschreibt.*

Damit werden je nach Fragestellung die Begriffe *Dienstgüte* und *Netzwerk-Performance* folgendermaßen verwendet:

Steht die Planung und Dimensionierung des Netzes im Vordergrund, so wird der Begriff *Netzwerk-Performance* verwendet, sonst der Begriff *Dienstgüte* oder *Quality of Service*.

Bei der Beschreibung von Messungen in dieser Arbeit wird auf das OSI-Schichtenmodell Bezug genommen. Dabei werden die Besonderheiten der TCP/IP-Protokollfamilie berücksichtigt. So wird die Implementierung oberhalb der Transport-Schicht (OSI-Schicht 4) als Anwendungsschicht bzw. Anwendungsebene bezeichnet.

2.1.4 Unterschiedliche Rollen von Anbietern und Nutzern in IP-Netzen

In den herkömmlichen Telekommunikationsnetzen gibt es im Wesentlichen eine Zweiteilung der Rollen – die Anbieter der Dienste auf der einen Seite und die Nutzer dieser Dienste auf der anderen. Dabei treten die Anbieter der Dienste gleichzeitig als Anbieter der Netzinfrastruktur auf. Dadurch sind die im Netz angebotenen Dienste fest an

die Netzinfrastruktur gekoppelt. Außerdem sind in herkömmlichen Telekommunikationsnetzen die Dienste häufig fest in der Netzinfrastruktur implementiert. Ein Beispiel dafür ist das *Integrated Services Digital Network (ISDN)* [ITU93]. Die ISDN-Dienste für die Teilnehmer sind in den Ortsvermittlungsstellen (OVSt) implementiert. Eine flächendeckende Einführung neuer Dienste erfordert damit Updates in allen OVSts. Da diese auch die Basisfunktionalität bereitstellen, sind solche Updates mit erhöhten Ausfallrisiken für die Basisdienste verbunden.

Im Internet sind die Rollen der Netzbetreiber und der Dienstanbieter voneinander getrennt. Somit findet man im Internet drei wesentliche Rollen vor:

1. Die *Netzbetreiber*, auch *Internet Service Provider (ISP)* genannt, stellen die Infrastruktur sowie die grundlegenden Dienste bereit. Dazu gehören unter anderem das Routing, die Einwahl in das Netz sowie die Authentifizierung der Teilnehmer.

2. Die *Dienstanbieter*, auch *Application Service Provider (ASP)* genannt, stellen unterschiedliche Dienste im Netz bereit. Durch die Entkopplung der Bereitstellung der Dienste von der Infrastruktur erfährt das Internet eine vielfach stärkere Dynamik in der Entwicklung neuer Dienste als die herkömmlichen Telekommunikationsnetze. Beispiele der von ASP angebotenen Dienste sind Web-Anwendungen, Mail-, News-Dienste sowie das Internet Telephony Service Providing (ITSP).

3. Die *Nutzer* stehen somit in einer Kunden-Beziehung sowohl den Netzbetreibern als auch den Dienstanbietern gegenüber. Wegen der Vielfalt der Dienste im Internet stehen die Nutzer in der Regel in einer Kunden-Beziehung mit mehreren Dienstanbietern.

Die dargestellte Dreiteilung der Rollen nimmt Einfluss auf die Betrachtung und Untersuchung der Dienstgüte je nach angenommener Rolle. Zu beachten ist, dass einzelne Einrichtungen und Personen je nach Problemstellung unterschiedliche Rollen annehmen können. So sind Anbieter von Diensten gleichzeitig auch Nutzer im Verhältnis zu den Internet Service Providern. Ebenso können Nutzer bestimmter Dienste wiederum Dienstanbieter anderen Nutzern gegenüber sein. Ein typisches Beispiel für solche doppelte Belegung der Rollen sind Betreiber von Suchmaschinen im World Wide Web. Diese sind *Nutzer* von vielen Web-Servern im Internet und gleichzeitig *Dienstanbieter* für die *Nutzer* dieser Suchmaschinen. Die unterschiedlichen Sichten auf Messungen je nach angenommener Rolle werden in Kapitel 3.1 vorgestellt.

2.2 Verteilung der Netzressourcen bei einer Paketvermittlung

Eine der wesentlichen funktionalen Eigenschaften des Internets ist die Paketvermittlung in Netzknoten. Dabei werden keine Ressourcen für die Verbindungen im Netz reserviert. Abbildung 2.1 zeigt schematisch den Vermittlungsmechanismus in einem IP-Vermittlungssystem[1] [Kur02, Kap 4.6]. Dabei werden Datenpakete beim Empfang im Netzknoten in einem Puffer – einer so genannten Input-Queue – zwischengespeichert. Im Gegensatz zur Leitungsvermittlung, die z.B. in ISDN-Netzen zum Einsatz kommt, wird die Quell- und Zielinformation in jedem einzelnen Datenpaket im IP-Header gespeichert. Anhand der Informationen im IP-Header wird festgelegt, auf welchem Interface das Datenpaket zum nächsten Netzknoten weitergeleitet wird. Zu den Vorteilen der

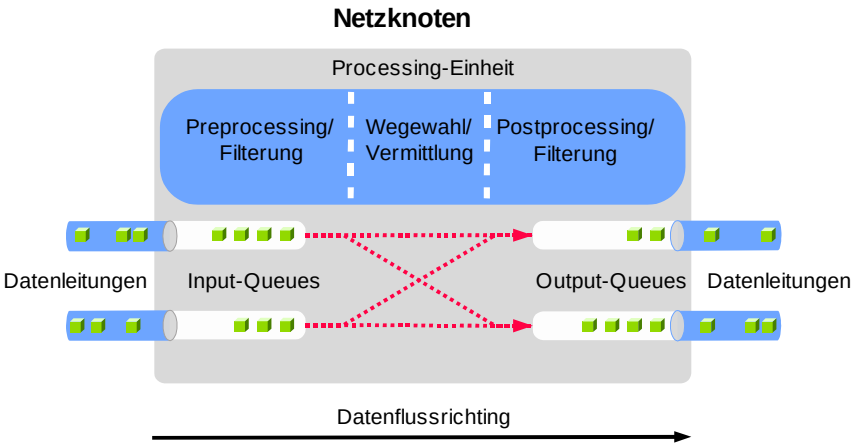

Abbildung 2.1: Prinzipskizze der Paketvermittlung in Netzknoten

Paketvermittlung gegenüber einer Leitungsvermittlung gehört zum einen ein schneller Vermittlungsvorgang und zum anderen die geringere Komplexität der Vermittlungssysteme z.B. gegenüber Vermittlungsstellen im ISDN. Dabei kann aber nicht sichergestellt werden, dass zu jedem Zeitpunkt der Ressourcenbedarf der Endsysteme vom Netz gedeckt werden kann. Ein weiterer Nachteil der Paketvermittlung ist der große Overhead bei der Datenübertragung.

Das Problem des großen Overheads kommt insbesondere bei Anwendungen mit geringer Länge der Nutzdaten zum Tragen. IP-Pakete werden in einen Protokollrahmen der Sicherungsschicht (Schicht 2 des OSI-Stacks) übertragen, bei dem wiederum Quell-

[1] Als Vermittlungssysteme in IP-Netzen agieren in erster Linie IP-Router, die eine Wegewahl treffen und anschließend die Datenpakete in Richtung Ziel vermitteln. Das Prinzip der Paketvermittlung trifft aber ebenso zu auf sonstige Komponenten im Netz, wie z.B. Switches und Access-Gateways. Aus diesem Grund werden im weiteren Verlauf der Arbeit die Komponenten zur Vermittlung von Datenpaketen in der Schicht 2 und 3 allgemein als Netzknoten bezeichnet.

und Zielinformationen in jedem Protokollrahmen übertragen werden. Ebenso werden Transportadressen im Protokollheader der Transportschicht übermittelt. Abbildung 2.2 zeigt beispielhaft den Aufbau eines Datenpakets und der darin enthaltenen Header für eine Sprachübertragung, die gemäß ITU-T G.729 [ITU96c] codiert und mit dem *Real-time Transport Protocol (RTP)* [Sch03] auf einem Ethernet-Link realisiert wird. Der

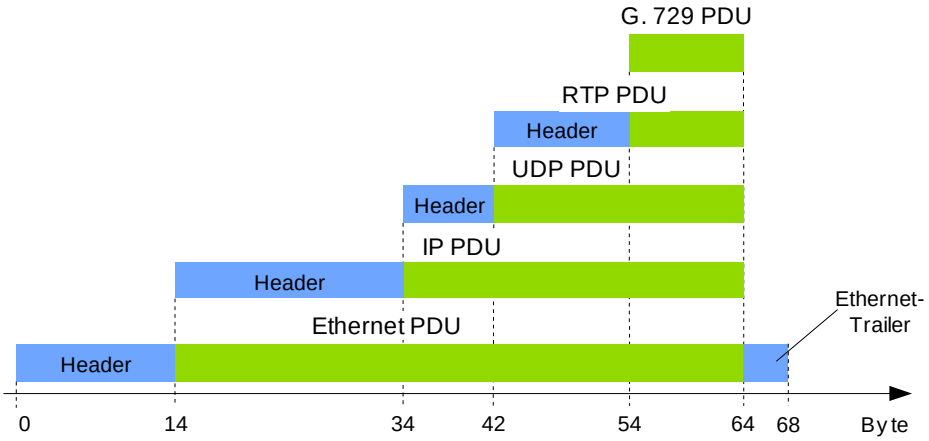

Abbildung 2.2: Overhead bei einer Sprachübertragung

Protokoll-Overhead der Schichten 2 bis 6 übersteigt die Menge der Nutzdaten um den Faktor $\frac{58}{10} = 5,8$. Dies muss bei der Planung von Netzwerk-Anwendungen bzw. bei der Dimensionierung der Netze berücksichtigt werden.

In [Sie00, S. 46ff.] wird gezeigt, dass ohne Zuhilfenahme gezielter Maßnahmen zur Kontrolle der Dienstgüte einzelne Datenströme bzw. einzelne Endsysteme gewollt oder ungewollt benachteiligt werden können. Die fehlende Reservierung und Steuerung der Belegung der Ressourcen kann sich in einer stark schwankenden Qualität der Dienste widerspiegeln. Kommen zu einem Zeitpunkt mehr Datenpakete in einem Netzknoten an als dieser verarbeiten kann bzw. als er auf die Ausgangsleitungen vermitteln kann, so verweilen die Datenpakete in einer Warteschlange, bis die Ausgangsleitung frei wird bzw. bis sonstige Ressourcen im Netzknoten – z.B. eine CPU oder eine Vermittlungseinheit – freigegeben werden. Damit steigt die Verweildauer der Datenpakete in den Warteschlangen. Ist die Warteschlange überfüllt, so werden die ankommenden Datenpakete verworfen. So entstandene Verzögerungen und Verluste von Datenpaketen beeinträchtigen die Anwendungen im Netz.

Die Netzanwendungen reagieren auf derartige Störungen sehr unterschiedlich. HTTP- sowie FTP-Anwendungen tolerieren beispielsweise wegen der Verwendung des Transport Control Protocols (TCP) hohe Paketverlustraten. Voice-over-IP, Videokonferenz-Dienste oder Datenbank-Anwendungen wiederum können bei steigenden Verzögerungen oder Paketverlustraten sowie bei stark schwankenden Laufzeiten von Datenpaketen

nicht mehr genutzt werden. Da die meisten Schwankungen der Laufzeiten ihre Ursache in der variablen Verzögerung der Datenpakete in den Warteschlangen der Netzknoten haben (siehe Kapitel 2.4.1), sind Mechanismen zur Kontrolle dieser Verzögerungen – so genannte Queueing-Mechanismen – für die Sicherung der Dienstgüte im Netz essentiell wichtig.

Um den Anforderungen unterschiedlicher Anwendungen an das Netz Rechnung zu tragen, wurden seit Anfang der 1990er Jahre von der IETF mehrere Ansätze für das Internet Protocol unternommen, die benötigte Dienstgüte im Internet zu steuern. Dabei werden die Ressourcen durch eine „gesteuerte Unfairness“ wichtigeren Anwendungen im Netz auf Kosten sonstiger Anwendungen zugeteilt. Diese Ansätze beinhalten aber keine direkten Mechanismen zur Steuerung der Verzögerung der Datenpakete in den Warteschlangen.

- Mit dem Ansatz der *Integrated Services (IntServ)* wurde versucht, für bestimmte Verbindungen Ressourcen im Netz zu reservieren und entsprechend Verbindungszustände dafür im Netz zu verwalten. Die Integrated-Services-Architektur sowie das dafür entwickelte *Resource Reservation Protocol (RSVP)* sind in [Bra94, Bra97] der IETF spezifiziert. Der IntServ-Ansatz hat sich aber nicht durchgesetzt und wird in dieser Arbeit deshalb nicht weiter berücksichtigt.

- In dem Ansatz der *Differentiated Services (DiffServ)* wurde versucht, eine gut skalierbare Lösung für die Sicherung der Dienstgüte in IP-Netzen zu installieren. Dabei wird eine gesonderte Behandlung unterschiedlicher Verkehrsklassen in den Netzknoten vorgenommen. Zwischen einzelnen Verkehrsklassen wird anhand einer Markierung im IP-Header – dem so genannten *Differentiated Services Codepoint (DSCP)* – unterschieden. Das Architekturmodell der Differentiated Services ist in RFC 2475 [Bla98] spezifiziert, die Verwendung des DSCP in RFC 2474 [Nic98].

  Der DiffServ-Ansatz findet zunehmend Anwendung. Bei der Durchführung von Messungen in IP-Netzen ist ggf. die unterschiedliche Behandlung einzelner Verkehrsklassen zu berücksichtigen. Werden bei der Durchführung von Messungen unterschiedliche Verkehrsprofile nachgebildet, so muss auch der Wert des DSCP nachgebildet werden (siehe Kapitel 3.2.3.1). Das Setzen des DSCP in den IP-Headern wird als *Markierung* der Datenpakete bezeichnet.

2.3 Verwendete Begriffe

Das Thema Messen in IP-Netzen ist relativ neu, daher werden einige Begriffe in Verbindung mit Netzmessungen diffus verwendet. Die Festlegung eindeutiger Begriffe und

Definitionen ist aber eine zwingende Voraussetzung für die Gewinnung von reproduzierbaren und übertragbaren Messergebnissen. Nachfolgend wird die Verwendung einiger Begriffe festgelegt.

In dieser Arbeit werden die Vorsätze ausschließlich in der im SI-System verankerten Bedeutung verwendet. Dies betrifft die Angabe von Datenmengen ebenso wie die Angabe von Übertragungsgeschwindigkeiten. Da ein Bit eine kleinste Informationseinheit bildet, werden keine Vorsätze für negative Potenzen verwendet. Die Informationsmenge wird in der Regel als Vielfaches von 8 Bits gespeichert, verarbeitet oder übertragen. Folglich wird die Informationsmenge in *Byte* angegeben. Die Übertragungsgeschwindigkeit wird dagegen in *Bit/s* ggf. mit entsprechenden Vorsätzen angegeben.

2.3.1 Einweg-Verzögerung

Die Metrik *Einweg-Verzögerung (engl. One-way Delay, OWD)* wurde von der IPPM Working Group definiert [Alm99a]. Als Einweg-Verzögerung wird die Zeit zwischen dem Aussenden des ersten Bits eines Datenpakets auf die Leitung beim Sender und dem Empfang des letzten Bits von der Datenleitung beim Empfänger definiert. Es ist zu beachten, dass die Messung der OWD eine endliche Zeit in Anspruch nimmt, wobei das Ergebnis wiederum eine Funktion der Zeit ist. Folglich ist bei der Angabe eines Wertes für diese Metrik die Festlegung des Bezugszeitpunktes relevant. Der Bezugszeitpunkt T für die Angabe der Einweg-Verzögerung ist die Zeit, an der das erste Bit vom Sender auf die Leitung gegeben wird.

Die Einweg-Verzögerung kann z.B. in einer aktiven Einweg-Messung bestimmt werden (siehe auch Abbildung 3.7). Dabei sind nach [Alm99a] für eine Messung folgende Parameter von Relevanz:

- der Bezugszeitpunkt T
- das verwendete Transport-Protokoll
- die Port-Nummer
- die Paketlänge
- die DSCP-Markierung

2.3.2 Umlauf-Verzögerung

Als *Round Trip Delay (RTD)* – auch *Round Trip Time (RTT)* genannt – zwischen zwei Endsystemen wird die Umlauf-Verzögerung eines Datenpakets bezeichnet. Die Umlauf-

Verzögerung ist die Zeitspanne zwischen dem Aussenden des ersten Bits eines Datenpakets und Empfang des letzten Bits eines Antwortpakets am Sender. Dabei muss das Antwortpaket unmittelbar nach Erhalt des Datenpakets vom Empfänger zurück ausgesendet werden. Der Bezugszeitpunkt T für die Angabe der Umlauf-Verzögerung ist wiederum der Zeitpunkt, an dem das erste Bit des Datenpakets auf die Leitung gegeben wird [Alm99c].

Die RTT kann entweder direkt (siehe auch Abbildung 3.4) oder aus der Messung der Einweg-Verzögerungen in entgegengesetzten Richtungen bestimmt werden. Im letzteren Fall ist auf den Bezugszeitpunkt T besonders zu achten. Wird zum Zeitpunkt T eine Einweg-Verzögerung D_h zwischen dem Quell- und Ziel-System gemessen, so muss als Bezugspunkt für die Messung der Einweg-Verzögerung in entgegengesetzter Richtung nicht der Zeitpunkt T sondern $T + D_h$ genommen werden. Beträgt also zum Zeitpunkt $T + D_h$ die Einweg-Verzögerung vom Ziel-System zur Quelle den Wert D_r, so ist die RTT zum Zeitpunkt T gleich $D_{RTT} = D_h + D_r$. Die Umlauf-Verzögerung wird als unendlich betrachtet, wenn das ausgesendete Datenpaket entweder am Ziel-System nicht angekommen ist oder auf dem Rückweg bei der Quelle nicht eintrifft. Die relevanten Parameter der Messung sind dieselben wie bei der Messung der Einweg-Verzögerung.

2.3.3 Jitter

Mit zunehmender Verbreitung zeitkritischer – vor allem multimedialer – Anwendungen im Internet wird der Maßgröße *Jitter* mehr und mehr Aufmerksamkeit gewidmet. Da dieser Begriff häufig mit unterschiedlicher Interpretation belegt wird, ist hier besonders auf eine korrekte Definition dieser Größe zu achten. Nachfolgend werden die unterschiedlichen Interpretationen von Jitter genannt und anschließend die Verwendung des Begriffs in dieser Arbeit festgelegt.

Allgemein dient der Begriff Jitter als Maß für die Abweichung des Zeitpunktes eines Ereignisses von einer vordefinierten Zeit. In Bezug auf eine digitale Datenübertragung definiert der US-Standard T1.523-2001 [T1.523] Jitter als eine kurzzeitige abrupte Abweichung markanter Zeitpunkte einer digitalen Übertragung von der idealen zeitlichen Lage. In dieser Arbeit wird Jitter im Wesentlichen in zwei unterschiedlichen Kontexten verwendet:

1. **Jitter im Kontext der Messung der Paketlaufzeiten.** Hierfür gibt es zwei häufig verwendete Definitionen. Die eine versteht unter Jitter die Abweichung der Laufzeit eines konkreten Datenpakets von der mittleren Laufzeit innerhalb eines betrachteten Zeitintervalls. Formal kann diese Definition folgendermaßen dargestellt werden:

Betrachtet seien N ausgesendete Datenpakete. Jedes sei mit der Ordnungsnummer $i,\ 1 \leq i \leq N$, gekennzeichnet. Die Aussendezeit des i-ten Datenpakets sei mit $t_{s,i}$ bezeichnet, die Empfangszeit als $t_{r,i}$. Die Aussendezeiten sowie die Empfangszeiten der Datenpakete 1 bis N seien bekannt, verlorene Pakete sollen dabei nicht berücksichtigt werden. Damit kann der Mittelwert m_D der Einweg-Verzögerungen folgendermaßen bestimmt werden:

$$m_D = \sum_{i=1}^{N} \frac{t_{r,i} - t_{s,i}}{N},\ 1 \leq i \leq N \tag{2.1}$$

Mit dem Mittelwert m_D aus (2.1) kann nun der Jitter wie folgt bestimmt werden:

$$J_i = (t_{r,i} - t_{s,i}) - m_D,\ 1 \leq i \leq N \tag{2.2}$$

Diese Definition des Jitters kursierte Ende der 90er Jahre in den Newsgroups und Mailing-Listen der IntServ Working Group der IETF. Ein wesentlicher Nachteil dieser Definition ist, dass der Jitter nur am Ende des betrachteten Zeitintervalls bestimmt werden kann.

Später, bei der Diskussion des DiffServ-Ansatzes, wurde eine neue Definition für den Jitter vorgeschlagen. Demnach wurde unter Jitter die Differenz der Laufzeit von zwei benachbarten Datenpaketen verstanden. Für die formale Definition des Jitters seien zwei direkt nacheinander ausgesendete Datenpakete betrachtet aus einer Gesamtmenge N ausgesendeter Datenpakete. Der Index des ersten betrachteten Datenpakets wird als $i-1$ bezeichnet, der des nachfolgenden Datenpakets i. Die Aussendezeiten der betrachteten Datenpakete $t_{s,i-1}$ und $t_{s,i}$ sowie die Empfangszeiten $t_{r,i-1}$ und $t_{r,i}$ seien bekannt. Damit lässt sich der Jitter des i-ten Datenpakets wie folgt definieren:

$$J_i = \left|(t_{r,i} - t_{s,i}) - (t_{r,i-1} - t_{s,i-1})\right|,\ 1 < i \leq N \tag{2.3}$$

Die Definition nach (2.3) hat den Vorteil, dass für die Berechnung von Jitter nur die Laufzeiten von zwei benachbarten Datenpaketen zu bestimmen sind. Wird das Datenpaket i nicht empfangen, so ist der Jitter J_i sowie J_{i+1} undefiniert.

Betrachtet man die Güte audiovisueller Anwendungen, so ist es für das flüssige Abspielen der Daten auf dem Ausgabegerät notwendig, dass die Datenpakete an den markanten Zeitpunkten – in der Regel zeitlich äquidistant – zum Abspielen zur Verfügung stehen. Haben zwei benachbarte Datenpakete unterschiedliche Laufzeiten, so wird die Anwendung gestört. Die Definition der Jitter nach (2.3) quantifiziert genau diese Störung der Anwendung. Die auf diese Weise quantifizierte Größe *Jitter* kann z.B. bei der Dimensionierung von Jitter-Buffern verwendet werden.

Von der IPPM wurde eine Metrik *„IP Packet Delay Variation“ (IPDV)* (siehe Tabelle 2.1) definiert, die ähnlich der nach (2.3) ist:

$$J_i = (t_{r,i} - t_{s,i}) - (t_{r,i-1} - t_{s,i-1}),\ 1 < i \leq N \tag{2.4}$$

Der Unterschied zur Gleichung (2.3) liegt in dem Verzicht auf den Betrag. Damit ist der Jitter eine vorzeichenbehaftete Größe, wobei das Vorzeichen angibt, ob ein Datenpaket verfrüht oder verspätet angekommen ist. Um eine Mehrdeutigkeit bei der Verwendung des Begriffs zu vermeiden, hat man bei der IPPM Working Group stattdessen den Begriff „Variation der Verzögerungen“ für die Definition nach (2.4) verwendet. Andere Forschungsgruppen (siehe z.B. [Low02]) verwenden dafür aber nach wie vor den Begriff Jitter. In der vorliegenden Arbeit wird der Begriff Jitter nach (2.3) verwendet.

2. **Jitter im Kontext der Betrachtung der Güte eines Senders.** Der Begriff *Jitter* ist im Allgemeinen nicht auf die Betrachtung der Laufzeiten im Netz beschränkt. Es sei beispielsweise ein Fall betrachtet, in dem eine Datenquelle in gleichen zeitlichen Abständen Datenpakete derselben Größe aussenden soll. Eine derartige Quelle wird als *Constant-Bit-Rate-Quelle (CBR-Quelle)* bezeichnet. Ist der Zeitpunkt für das Aussenden eines Datenpakets bekannt, so können die Zeitpunkte berechnet werden, an welchen alle anderen Datenpakete versendet werden sollen. Eine CBR-Datenquelle kann durch zwei Parameter – die Datenrate r und die Paketlänge S – beschrieben werden. Wird der Zeitpunkt für das Aussenden des ersten Datenpakets als t_1 bezeichnet, kann die Aussendezeit für das Aussenden des i-ten Datenpakets wie folgt bestimmt werden:

$$t_i = t_1 + (i-1) \cdot \frac{S\ [Byte] \cdot 8\ [Bit/Byte]}{r\ [Bit/s]} \tag{2.5}$$

Die Zeitpunkte $t_i,\ 1 \leq i \leq N$ können nach dem oben genannten US-Standard T1.523-2001 als markante Zeitpunkte einer digitalen Datenübertragung betrachtet werden. In der Realität werden aber durch im System aufgetretene Unsicherheiten die Datenpakete jeweils zu den Zeitpunkten $\tau_1, \tau_2, \cdots, \tau_N$ ausgesendet. Damit kann die Differenz zwischen den jeweiligen Zeitpunkten, an denen die Datenpakete im Idealfall ausgesendet werden sollten und denen des tatsächlichen Absendens, ebenso als *Jitter* bezeichnet werden.

$$J_i = |\tau_i - t_i| \tag{2.6}$$

In diesem Fall zeigt der Jitter, wie genau die Datenquelle sich an das vorgegebene Quellenmodell hält. Diese Art von Jittern wird in der Arbeit als *Sender-Jitter* bezeichnet. Steigende Werte der Sender-Jitter bedeuten eine steigende Abweichung

der realen Datenquelle vom idealen Quellenmodell. Das dargestellte Modell für Sender-Jitter kann auf Datenquellen beliebiger Art angewendet werden, falls die gewünschten Aussendezeiten bekannt sind. Werden die Ankunftszeiten der Datenpakete am Empfänger betrachtet, so werden die Abweichungen von der idealen zeitlichen Lage entsprechend als *Empfänger-Jitter* bezeichnet.

Es ist zu beachten, dass im Kontext der Betrachtung der Paketlaufzeiten der Jitter die Güte eines Übertragungssystems charakterisiert. Der Sender-Jitter charakterisiert vielmehr die Güte eines Datensenders.

2.3.4 Durchsatz

Bei der Betrachtung der Netzanwendungen ist die Menge der Daten, die innerhalb einer Zeitspanne ΔT zwischen zwei Endsystemen übertragen werden kann, zu berücksichtigen. Die Anzahl an Bytes, die über eine Zeitspanne ΔT übertragen werden kann, wird als *Durchsatz (engl. Bulk Transfer Capacity, BTC)* bezeichnet. Mit der Bezeichnung B für die übertragene Informationsmenge sowie R_{BTC} für den Durchsatz kann dieser wie folgt berechnet werden:

$$R_{BTC} = \frac{B}{\Delta T} \tag{2.7}$$

Für den Durchsatz existiert zurzeit keine von der IPPM Working Group definierte Metrik. Es gab in der Vergangenheit den Versuch, die *Bulk Transfer Capacity Metric* zu definieren. Bei der Definition konnten aber nicht alle Nebenbedingungen, welche diese Größe beeinflussen, genannt werden [Mat01]. Das Problem der Nennung der Nebenbedingungen liegt darin, dass große Datentransfers, bei welchen die BTC von besonderer Bedeutung ist, überwiegend über TCP realisiert werden. Dabei hängt aber der Durchsatz von internen Variablen der TCP-Implementierung ab. Der erreichte Durchsatz ist somit von der konkreten Implementierung des TCP-Stacks abhängig.

2.3.5 Paketverlust

Eine weitere wesentliche Messgröße ist der Paketverlust. Ein Datenpaket gilt als verloren gegangen, wenn es vom Sender ausgesendet, beim Empfänger aber nicht empfangen wird. Eine so definierte Größe wird als *Einweg-Paketverlust (engl. One-way Packet Loss*) [Alm99b] bezeichnet. Der Einweg-Paketverlust entspricht einer unendlichen Einweg-Verzögerung. In der Praxis wird bei der Bestimmung eines Paketverlustes mit einem Timeout für den Empfang von Datenpaketen gearbeitet. Werden in

mehreren Messungen unterschiedliche Werte des Timeouts verwendet, so können unterschiedliche Ergebnisse erreicht werden. Somit gehört zur Beschreibung einer Messumgebung die Angabe des verwendeten Wertes für das Timeout. Bei vielen Anwendungen – z.B. bei solchen, die über TCP realisiert werden – sind diese Werte fest in der Implementierung des TCP-Protokolls verankert. Neben dem Einweg-Paketverlust kann der *Umlauf-Paketverlust* betrachtet werden. Dabei gilt ein Datenpaket als verloren, wenn es entweder auf dem Weg vom Sender zum Empfänger oder auf dem Weg zurück verloren gegangen ist.

Es sind unter anderem auch die vom Paketverlust abgeleiteten Größen von Interesse. Eine solche Größe ist dabei die *Paketverlustrate*. Diese quantifiziert die Anzahl verloren gegangener Datenpakete innerhalb eines Beobachtungszeitraumes in Relation zur Anzahl ausgesendeter Datenpakete. Die *Verlustrate* beschreibt die Relation verloren gegangener zu zur ausgesendeten Informationsmenge in Bytes. Außerdem sind so genannte *Paketverlust-Muster* von Bedeutung [Koo02]. Dabei wird die Anzahl der unmittelbar nacheinander verloren gegangenen Pakete, bzw. die Zeitdauer, in welcher alle Datenpakete verloren gegangen sind, sowie die Dauer der verlustfreien Übertragung von Datenpaketen betrachtet. Die Anzahl der aufeinander folgenden verloren gegangenen Datenpakete wird als *Paketverlust-Distanz* bezeichnet. Die Zeitspanne, in der alle Datenpakete verloren gegangen sind, wird *Paketverlust-Periode* genannt. Häufig wird für die beiden Größen etwas allgemeiner der Begriff *Burstiness* der Paketverluste verwendet, wobei mit starker Burstiness der Paketverluste große Paketverlust-Perioden bzw. -Distanzen verstanden werden. Diese ist aber keine quantitative, sondern eine qualitative Größe.

2.3.6 Definition eines Datenstroms

In Verbindung mit der Untersuchung der Netzwerk-Performance wird häufig der Begriff *Datenstrom (engl. Flow)* verwendet. Darunter wird sinngemäß eine geordnete Folge von Datenpaketen verstanden, die in einem logischen Zusammenhang zueinander stehen.

Für die formale Definition eines TCP- bzw. UDP-Datenstroms sei ein Tupel $\mathcal{T}_i$ von Parametern betrachtet. Diese Parameter sind in Tabelle 2.2 dargestellt. Zur gleichen Zeit kann im Netz nur ein Datenstrom durch dasselbe $\mathcal{T}_i$ beschrieben werden. Über einen längeren Zeitraum können mehrere Datenströme, beschrieben durch dasselbe Tupel $\mathcal{T}_i$ beobachtet werden. Damit wird ein weiteres Kriterium benötigt, um Datenpakete einem Datenstrom eineindeutig zuzuordnen.

TCP hat eine verbindungsorientierte Semantik. Bevor eine Datenübertragung stattfindet, wird eine TCP-Verbindung explizit aufgebaut. Ist eine Datenübertragung abgeschlossen, so wird die Transport-Verbindung explizit beendet [Ste94a, Kap. 18]. Somit

Identifikation	Parameter
Src-Addr	Absender-IP-Adresse eines Datenpakets
Dest-Addr	Empfänger-IP-Adresse eines Datenpakets
Src-Port	Port-Adresse des Senders
Dest-Port	Port-Adresse des Empfängers
Proto	Verwendetes Transport-Protokoll

Tabelle 2.2: Identifikation eines Datenstroms

kann ein TCP-Datenstrom durch die Zugehörigkeit von zwei Datenpaketen zu demselben Datenstrom wie folgt definiert werden:

Definition 2.3 *Zwei TCP-Datenpakete gehören zu demselben Datenstrom, wenn diese durch dasselbe Tupel $\mathcal{T}_i$ beschrieben sind und in der Zeit zwischen der Beobachtung dieser Datenpakete kein erfolgreicher Verbindungsabbau für eine Verbindung, stattgefunden hat, zu dem der Datenstrom $\mathcal{T}_i$ gehört.*

Zu beachten ist, dass nach der Definition 2.3 jede TCP-Verbindung aus zwei entgegengerichteten TCP-Datenströmen besteht. Solche Richtungstrennung ist für die Betrachtung der Netzwerk-Performance sinnvoll, da sowohl die Charakteristik des Datenstroms als auch die Auswirkung der Netzwerk-Performance auf die Anwendung in der Hin- und Rückrichtung unterschiedlich sind.

UDP hat keine verbindungsorientierte Semantik. Trotzdem arbeiten viele UDP-basierte Anwendungen, wie z.B. VoIP und TFTP [Sol92], verbindungsorientiert. Hier kann die Zuordnung der Datenpakete zu einem Datenstrom in den Ebenen 3 und 4 des OSI-Stacks nur über den zeitlichen Abstand von zwei benachbarten Datenpaketen mit demselben Tupel $\mathcal{T}_i$ stattfinden. Damit wird ein UDP-Datenstrom wie folgt definiert:

Definition 2.4 *Zwei UDP-Datenpakete gehören zum selben UDP-Datenstrom, wenn diese durch dasselbe Tupel $\mathcal{T}_i$ beschrieben werden und zwischen den Zeitpunkten der Beobachtung dieser Datenpakete kein Zeitintervall $\Delta t \geq T_{Timeout}$ existiert, in dem nicht mindestens ein Datenpaket, beschrieben durch dasselbe Tupel $\mathcal{T}_i$, beobachtet wurde.*

Es existiert kein wohldefinierter Wert für den Timer $T_{Timeout}$. Ein geeigneter Wert des Timers lässt sich für einige Klassen von Anwendungen empirisch ermitteln. Die Gesichtspunkte für die Ermittlung des Timers $T_{Timeout}$ für multimediale Anwendungen werden in Anhang A.1 vorgestellt.

2.4 Messgrößen

In Kapitel 2.1 wurde bereits festgelegt, dass in dieser Arbeit lediglich technische Aspekte der Dienstgüte betrachtet werden. In Kapitel 2.3 wurden Definitionen von einigen Messgrößen bzw. Metriken gegeben. Nachfolgend wird der Einfluss einzelner Parameter der Dienstgüte auf die Anwendungen diskutiert.

Um einen Überblick zu bekommen, ist eine Auswahl von Anwendungen mit den damit verbundenen Protokollen in Tabelle 2.3 zusammengefasst.

Anwendungen	**Relevante Protokolle bzw. Middleware**	**qualitätsrelevante Merkmale**
Datentransfer-Anwendungen	HTTP, FTP, SMTP	kurzzeitig hoher Durchsatz, variable Umlauf-Verzögerungen
IP-Telefonie, Medienströme	G.711, G.723.1, G.726, G.729	geringe Datenrate, geringe Einweg-Verzögerungen, sehr geringe Jitter, geringe Paketlängen
Videokonferenzen, Medienströme	G.261, G.263, MPEG2, MPEG4	hohe Datenrate, geringe Einweg-Verzögerungen, sehr geringe Jitter, variable Paketlängen
Videostreaming-Anwendungen	RealVideo, MPEG2, MPEG4, Windows Media	hohe Datenrate, geringe Jitter, überwiegend große Pakete
Signalisierung der Multimedia-Anwendungen	H.323, SIP, MGCP, H.225, H.245	geringe Paketverluste, geringe bis mittlere Paketlängen, variable Datenrate
Grid-Anwendungen	Globus, Unicore, Condor	geringe Umlauf-Verzögerungen, sehr hoher Durchsatz
Netz-Spiele	Quake3, proprietäre Protokolle	sehr geringe Umlauf-Verzögerungen, geringe Paketverluste, kleine Datenpakete
Datenbanken	SQL, ODBC, proprietäre Protokolle	geringe Paketverluste, hoher Durchsatz

Fortsetzung auf der nächsten Seite

		Fortsetzung, (siehe vorherige Seite)
Netz-Management	SNMP, CMIP, ICMP, proprietäre Protokolle	geringe Latenzen, geringe Verzögerungen
Alarm-Systeme	ICMP, proprietäre Protokolle	sehr geringe Latenzen, geringe Paketverluste

Tabelle 2.3: Anwendungen und deren qualitätsrelevanten Merkmale

In dieser Arbeit wird der Fokus auf die Betrachtung der Aspekte der Dienstgüte multimedialer Anwendungen am Beispiel der IP-Telefonie und der Videokonferenzen gelegt. Dieser Betrachtung liegen Untersuchungen zugrunde, die am derzeitigen Lehrgebiet Rechnernetze in den letzten fünf Jahren durchgeführt worden sind. Wegen der zunehmenden Bedeutung von Grid-Anwendungen werden deren besonderen Anforderungen in die Betrachtungen mit einbezogen.

2.4.1 Relevante Messgrößen

Die wesentlichen Messgrößen, welche die Qualität der in Tabelle 2.3 zusammengefassten Anwendungen beeinflussen, sind Verzögerungen, Jitter, Paketverluste sowie Durchsatz. Nachfolgend wird die Auswirkung dieser Parameter auf die Anwendungen diskutiert.

2.4.1.1 Paketverzögerung

Die Metriken Einweg-Verzögerung und Umlauf-Verzögerung wurden bereits in den Kapiteln 2.3.1 und 2.3.2 definiert. Die Kommunikation im Internet baut überwiegend auf einem Client-Server-Modell auf. Dabei sendet der Client eine Anfrage (Request) an den Server und wartet auf eine Antwort (Response). Die Reaktionszeiten – in welche die Paketverzögerungen in der Hin- und Rückrichtung einfließen – beeinflussen damit die wahrgenommene Qualität der Anwendungen. Für das World Wide Web werden die Komponenten, welche die wahrgenommene Güte der Anwendung beeinträchtigen, in [Cha01] vorgestellt.

Bei den auf der Client-Server-Architektur basierenden Anwendungen ist die Charakteristik der Kommunikation asymmetrisch. Somit ist auch der Einfluss der Verzögerung je nach Richtung unterschiedlich. Bei Medienströmen multimedialer Anwendungen ist die Kommunikation in beide Richtungen voneinander entkoppelt, somit ist im Allgemeinen die Einweg-Verzögerung von besonderem Interesse. Kann die Einweg-Verzögerung für die Hin- und Rückrichtung bestimmt werden, so kann die Umlauf-Verzögerung mit der in Kapitel 2.3.2 vorgestellten Methode berechnet werden. Eine eindeutige Bestimmung der Einweg-Verzögerung aus der Umlauf-Verzögerung ist jedoch nicht möglich. Es ist

aber eine Abschätzung der Einweg-Verzögerung aus der Umlauf-Verzögerung möglich (siehe Kapitel 3.2.4.1).

2.4.1.2 Jitter

Für ein flüssiges Abspielen von Sprach- und Bilddaten multimedialer Anwendungen ist es essentiell wichtig, dass die empfangenen Daten zum richtigen Zeitpunkt dem Ausgabegerät zur Verfügung stehen. Werden die benötigten zeitlichen Abstände zwischen einzelnen Bild- bzw. Sprach-Segmenten nicht eingehalten, so wird das entsprechende Signal verzerrt ausgegeben. Werden die empfangenen Segmente direkt an den Decoder gegeben, so kann die Bedingung der Äquidistanz nur dann erfüllt werden, wenn die Paketlaufzeiten im Netz konstant sind. In der Realität treten im Netz immer Jitter auf. Um eine Äquidistanz der Datenpakete trotz Jitter im Netz sicherzustellen, werden die empfangenen Daten in anwendungsinternen Buffern – so genannten Jitter-Buffern – zwischengespeichert. Zur Verdeutlichung ist der Empfang und das Abspielen von einem Audiostrom in Abbildung 2.3 vereinfacht dargestellt.

Abbildung 2.3: Stream-Service multimedialer Anwendungen

Beispiele der Auswirkung der Jitter auf die Sprachübertragung in IP-Netzen mit und ohne Verwendung von Jitter-Buffern sind in [Göh01] vorgestellt. In [Log01] wird durch Beobachtungen in einem IP-Netz gezeigt, dass 89,9 % der so genannten Buffer-Underruns – also der verspäteten Ankunft von Datenpaketen in einen Jitter-Buffer – durch Jitter im Netz verursacht sind. Die Definition des Jitters nach (2.3) kann für die Dimensionierung der Jitter-Buffer verwendet werden.

2.4.1.3 Durchsatz

Neben einer Vielzahl an so genannten elastischen Anwendungen, welche ihre Datenrate dynamisch an die verfügbaren Ressourcen im Netz anpassen, gibt es eine Reihe von Anwendungen, welche eine vorgegebene mittlere verfügbare Datenrate im Netz zwingend benötigen. Steht diese nicht zur Verfügung, so sinkt die Qualität der Anwendung oder sie kommt gänzlich zum Erliegen. Hieraus ergibt sich der Bedarf nach der Messung des Durchsatzes (BTC). Neben multimedialen Anwendungen ist die Bestimmung der BTC

für Grid-Anwendungen von besonderer Relevanz [Low02]. In Grid-Szenarien wird die Abschätzung der BTC benötigt, damit eine Entscheidung getroffen werden kann, von welchem Ort die benötigte Information abgerufen wird. In diesem Zusammenhang ist speziell die Vergleichbarkeit von Messergebnissen von besonderer Bedeutung.

2.4.1.4 Paketverlust

Hinsichtlich der Reaktion auf Paketverluste können die Anwendungen in IP-Netzen in zwei Klassen eingeteilt werden: in verlustsensitive und verzögerungssensitive Anwendungen.

Zu der ersten Klasse gehören Datentransfer-, Grid-, Netzmanagement- sowie Datenbank-Anwendungen (siehe Tabelle 2.3). Die Verlustfreiheit wird durch die Fehlerkorrektur im Transport-Protokoll (wie z.B. bei TCP) oder in der Anwendung (z.B. bei UDP-basierten Anwendungen wie SIP- oder DNS-Anwendungen [Pos84, Ros02]) erreicht. Bei solchen Anwendungen werden Paketverluste in der Netzwerk-Ebene als Verzögerungen auf der Anwendungsebene bemerkbar. Das Timeout für den Empfang einer Antwort auf einen Request liegt bei derartigen Anwendungen im Bereich von wenigen Sekunden (z.B. 5 s bei *nslookup* [Che97]). Bei solchen Anwendungen machen sich folglich Paketverluste durch lange Wartezeiten bemerkbar.

Bei verzögerungssensitiven Anwendungen ist es günstiger, auf ausgebliebene Information zu verzichten als auf die Wiederholung der Datenpakete zu warten. Zu solchen Anwendungen gehört vor allem die multimediale Dialog-Kommunikation. Ergebnisse von Untersuchungen der Auswirkung von Paketverlusten auf unterschiedlich codierte Sprachströme sind in [Göh01] vorgestellt. Der Einfluss von Paketverlusten auf die Güte von Video-Anwendungen wird in [Hei01] vorgestellt.

Der Bedarf nach der Betrachtung der Burstiness der Paketverluste wird in [Log01] gezeigt. Dort werden Ergebnisse von Langzeit-Messungen im US-amerikanischen Netz des Internet Service Providers UUNET vorgestellt. Diese belegen, dass ca. 30 % der Paketverluste nicht einzeln, sondern in Bursts auftreten. Die Annahme, dass vor allem die Güte von Anwendungen der audiovisuellen Kommunikation nicht nur von der Paketverlustrate, sondern auch von der Burstiness abhängt, ist nahe liegend und wird häufig bei der Untersuchung der Dienstgüte getroffen. In den Ergebnissen aus [Göh01, Hei01] wird diese Annahme indirekt bestätigt. Es fehlen jedoch systematische Untersuchungen, welche quantitative Zusammenhänge zwischen den Metriken der Paketverlust-Muster und der Beeinträchtigung von Anwendungen deutlich machen.

2.4.2 Interpretation der Messergebnisse

Bei Messungen in IP-Netzen wird in der Regel eine große Anzahl an Probepaketen ausgewertet. Häufig beinhalten Messreihen mehrere Millionen Messpunkte. Damit wird dann die Dienstgüte oder die Netzwerk-Performance mit statistischen Größen beschrieben.

Nachfolgend werden die Größen vorgestellt, welche die Messergebnisse quantifizieren. Die Anforderungen an die Werkzeuge für die Auswertung der Ergebnisse werden später in Kapitel 2.6.4 beschrieben.

2.4.2.1 Mittelwert

Der Mittelwert kann in erster Näherung die Eigenschaft einer Strecke oder einer Netzkomponente quantifizieren – z.B. die mittlere Laufzeit der Datenpakete oder die mittlere Datenrate auf einem Link. In Verbindung mit weiteren statistischen Größen können genauere Aussagen über die untersuchte Größe getroffen werden.

2.4.2.2 Extremwerte

Die Extremwerte einer Messreihe – das Minimum und das Maximum – können hilfreich sein, um Grenzwerte, die durch physische Gegebenheiten der Netze bedingt sind, aufzudecken. Als Beispiel sei eine Messung der Umlauf-Verzögerung im Netz betrachtet. Weist die Messung ausreichend viele Messpunkte (Samples) auf, so kann davon ausgegangen werden, dass das Minimum dieser Messwerte die untere Grenze der Verzögerung – verursacht durch die Serialisierungszeit (siehe Kapitel 2.4.3) sowie durch die Ausbreitungsgeschwindigkeit der Signale im Netz – repräsentiert. Unter bestimmten Randbedingungen kann anschließend aus dem Maximum der Umlauf-Verzögerung die Größe der Warteschlangen im System bestimmt werden (siehe Kapitel 5.4).

2.4.2.3 Standardabweichung und die Tschebyscheffsche Ungleichung

Eine weitere wichtige statistische Größe ist die Standardabweichung. Die Berechnungsvorschrift für die Standardabweichung σ für eine Messreihe mit dem Mittelwert m ist in Gleichung (2.8) dargestellt.

$$\sigma = \sqrt{\frac{1}{N} \cdot \sum_{i=1}^{N} (x_i - m)^2} \tag{2.8}$$

Die Standardabweichung ist ein Maß für die Streuung der Werte um den Mittelwert. Für die Berechnung der Wahrscheinlichkeit $p(\epsilon)$, dass ein Messwert nicht mehr als um einen bestimmten Wert ϵ vom Mittelwert m abweicht, kann beispielsweise die Tschebyscheffsche Ungleichung verwendet werden (siehe [Mer77, S. 42 ff.]). Diese lautet wie folgt:

$$p(\epsilon) \leq \frac{\sigma^2}{\epsilon^2} \tag{2.9}$$

Der Vorteil dieser Ungleichung ist, dass sie keine spezielle Verteilung der Messwerte voraussetzt. Es existieren weitere Verfahren, welche unter Annahme eines bestimmten statistischen Modells der Verteilung der Zufallsvariablen engere Grenzen für den Fehler abschätzen lassen. Entspricht die Streuung der Messergebnisse z.B einer Normalverteilung, so kann eine wesentlich bessere Abschätzung der Wahrscheinlichkeit als mit der Tschebyscheffschen Ungleichung gemacht werden (siehe [Mer77, S. 44 ff.]).

2.4.2.4 Verteilungsfunktionen der Messgrößen

Eine weitere Darstellungsmöglichkeit der Messergebnisse bieten statistische Verteilungen, insbesondere die Verteilungsfunktion (*Cumulative Distribution Function – CDF*). Die Berechnungsvorschrift für die Verteilungsfunktion einer Reihe von Messergebnissen x_i lautet:

$$P(x \leq X) = \sum_{x_i \leq X} P(x = x_i) \tag{2.10}$$

Wobei $P(x = x_i)$ die Häufigkeit bzw. die Wahrscheinlichkeit des Auftretens des Wertes x_i darstellt. Aus der CDF lässt sich die Häufigkeit für das Über- bzw. Unterschreiten eines bestimmten Grenzwertes direkt ablesen. Es existieren einige abgewandelte Formen der Verteilungsfunktion – z.B. die *Complementary Cumulative Distribution Function (CCDF)* und die *Wahrscheinlichkeitsverteilung, englisch Probability Distribution Function (PDF).*

2.4.3 Komponenten der Verzögerung der Datenpakete im Netz

Die Betrachtungen in Kapitel 2.4.1 zeigen, dass die Paketverzögerungen bzw. daraus abgeleitete Größen, wie z.B. Jitter und Paketverlust, die Güte der Anwendungen beeinträchtigen. Speziell bei der Abschätzung der Fehler für eine Messung von Paketverzögerungen ist es hilfreich, die Ursachen für die Verzögerungen festzustellen. Dafür werden in Abbildung 2.4 einzelne Komponenten der Verzögerung als Blockschaltbild dargestellt.

Mit t_s ist die Serialisierungszeit zusammengefasst. Dies ist die Zeitspanne zwischen dem Aussenden des ersten und des letzten Bits auf die physische Leitung. Eine Serialisie-

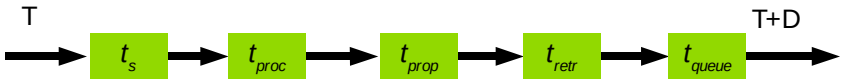

Abbildung 2.4: Blockschaltbild der Verzögerungskomponenten einer Datenübertragung

rungszeit t_s wird ebenfalls auf der Empfängerseite benötigt, um das Datenpaket von der Leitung in den Puffer einzulesen. Die Serialisierungszeit muss jeweils einmal für jede Verbindung zwischen jeweils zwei Knoten berechnet werden. Um dies zu verdeutlichen, soll beispielhaft ein Fall betrachtet werden, in dem ein Sender und ein Empfänger über eine unendlich kurze Leitung mit der Übertragungsrate R miteinander verbunden sind. Da die Leitung unendlich kurz ist, trifft das Datenpaket sofort nach dem Verlassen des Senders beim Empfänger ein. Somit ist zum Zeitpunkt, an dem das zweite Bit beim Sender auf die Leitung gegeben wird, das erste bereits im Speicher des Empfängers aufgenommen worden. Somit ist auch das gesamte Datenpaket, sobald es den Sender verlassen hat, im Buffer des Empfängers gespeichert. Die Serialisierungszeit ist offensichtlich direkt proportional zur Paketlänge und umgekehrt proportional zur Link-Datenrate. Mit der Paketlänge S in Byte und der Link-Datenrate R in Bit/s kann t_s wie folgt berechnet werden:

$$t_s = \frac{8 \cdot S}{R} \tag{2.11}$$

Auf einem Fast-Ethernet-Link (100 Mbit/s) beträgt die Serialisierungszeit eines Ethernet-Frames der Größe 1.500 Byte etwa 116 µsec.

In der Processing-Zeit t_{proc} sind die Verarbeitungszeiten in den Netzknoten zusammengefasst. Die Processing-Zeit beinhaltet die Behandlung der Pakete innerhalb der Netzknoten nach dem Verlassen der Input-Queue und vor dem Eintritt in die Output-Queue. Zum Processing gehört unter anderem die Wegewahl in den Routern, Berechnung der Prüfsummen sowie der Durchlauf durch bestimmte Paketfilter. An dieser Stelle wird kein Unterschied zwischen Netzkomponenten, wie z.B. Router oder Switches, und den Endsystemen gemacht, da die grundsätzliche Funktionsweise in beiden Fällen dieselbe ist.

Die Verarbeitungszeit ist in erster Linie von der Konfiguration sowie von der augenblicklichen Auslastung des Systems abhängig. Sie ist unabhängig von der Paketlänge. Bis auf Grenzfälle, in welchen die Netzkomponenten überlastet sind, kann diese Verzögerungskomponente als annähernd konstant betrachtet werden. Außerdem können moderne Netzknoten mehrere hunderttausend bis mehrere Millionen Datenpakete pro Sekunde im Dauerbetrieb verarbeiten. So kann z.B. der Router *Cisco 7500* bis zu 500.000 Datenpakete pro Sekunde zwischen einem Eingangs- und einem Ausgangs-Interface vermitteln [Cis04a]. Ethernet-Switches haben eine um 2 bis 3 Größenordnungen höhere

Performance. Ein Switch *Catalyst 6500* kann beispielsweise bis zu 500.000.000 Datenpakete pro Sekunde vermitteln [Cisc04b].

In Tabelle 2.4 sind maximale Paketraten und entsprechend minimale Paketabstände auf Fast-Ethernet- und Gigabit-Ethernet-Links dargestellt. Der minimale Paketabstand ergibt sich aus den in den Standards IEEE 802.3u und IEEE 802.3ab [IEEE98, IEEE99] festgelegten minimalen Rahmengrößen zuzüglich des Interframe-Gaps (12 Bytes), geteilt durch die Link-Datenrate. Der minimale Paketabstand ist der Kehrwert der maximalen Paketrate. Die maximale Paketrate entspricht dem Kehrwert des minimalen Paketabstands. Der minimale Abstand zwischen zwei Datenpaketen auf einem Interface ist viel größer als die Verarbeitungszeit dieser in einem leistungsfähigen Prozessor. Im Normalfall bestimmt also nicht die Prozessor-Leistung, sondern die maximale Paketrate den minimalen Abstand zwischen zwei Datenpaketen. Somit kann für richtig dimensionierte Systeme für die Komponente t_{proc} der minimale Abstand zwischen zwei Datenpaketen (letzte Spalte der Tabelle 2.4) als obere Grenze angenommen werden.

Protokoll	**Bezeichnung**	**minimale Rahmengröße [Byte]**	**maximale Paketrate [Pakete/s]**	**minimaler Paketabstand [µs]**
IEEE 802.3u	Fast Ethernet	72	148.809	6, 72
IEEE 802.3ab	Gigabit Ethernet	419	294.811	3, 39

Tabelle 2.4: Maximale Paketraten auf Fast- und Gigabit-Ethernet-Verbindungen

Wie oben erwähnt, gilt diese Abschätzung der Verarbeitungszeit nur für Systeme, in welchen der Prozessor nicht unterdimensioniert ist. Im Einzelfall ist dies ggf. zu prüfen. In Kapitel 5.4 wird eine Abschätzung der Verarbeitungszeit t_{proc} für ein System gemacht, in dem der Netzwerk-Prozessor die limitierende Komponente darstellt.

Die Fortpflanzungsverzögerung t_{prop} der Datenpakete ist von der Länge und der Physik der Verbindungswege zwischen den Kommunikationsendpunkten abhängig. Diese ist zeitinvariant, abgesehen von Routing-Änderungen im Netz, welche selten in Relation zu den üblichen Verbindungsdauern im Internet auftreten. Tabelle 2.5 fasst die Näherungswerte der Ausbreitungsgeschwindigkeiten sowie Ausbreitungszeiten der Signale in unterschiedlichen Medien zusammen.

In [Sub01, Car03] vorgestellte Untersuchungen der Topologie des Internets schätzen eine maximale Ausbreitungsverzögerung im Internet zwischen zwei Punkten auf terrestrischen Strecken von 200 ms ab. Dies entspricht einer maximalen Entfernung von zwei Punkten im Internet von etwa dem 1,7-fachen des halben Erdumfangs.

Medium	Ausbreitungs-geschwindigkeit [km/s]	Ausbreitungs-zeit [µs/km]
Luft, Raum	300.000	3, 33
Kupfer	210.000	4, 8
Glasfaser	180.000	5, 6

Tabelle 2.5: Ausbreitungsgeschwindigkeit der Signale in unterschiedlichen Medien

Die genannten Untersuchungen basieren unter anderem auf Daten aus Langzeit-Messungen, die von der Organisation CAIDA [CAI04] rund um den Globus gesammelt werden.

Weitere Verzögerungen können durch Wiederholungen (Retransmissions) von Rahmen der Sicherungsschicht auftreten. Diese sind in Abbildung 2.4 als t_{retr} eingetragen. Treten Retransmissions in der Sicherungsschicht auf, so kann t_{retr} vielfach größer sein als die Ausbreitungsverzögerung. Damit kann t_{retr} einige Millisekunden bis mehrere hundert Millisekunden groß werden.

Die Warteschlangen-Verzögerung t_{queue} ist die Verzögerung der Datenpakete in den Input- und Output-Queues der Netzknoten. In den Input-Queues werden die Pakete zwischengespeichert, bevor sie vom Netzwerk-Prozessor verarbeitet werden (siehe Abbildung 2.1). Hier entstehen signifikante Verzögerungen nur im Falle einer zu hohen Belastung des Prozessors. Ein derartiger Betriebszustand deutet auf eine falsche Dimensionierung der Netzressourcen hin.

In den Output-Queues werden Datenpakete gespeichert, bevor diese auf die Ausgangsleitung gegeben werden. Werden in einer Zeitspanne ΔT vom Prozessor mehr Datenpakete in eine Output-Queue gestellt als das Interface in dieser Zeit mit der Leitungsgeschwindigkeit aussenden kann, so werden diese Datenpakete in der Output-Queue gestaut. Die Verzögerung der Datenpakete ist dabei eine Funktion der Link-Datenrate des Interfaces sowie des aktuellen Verkehrsaufkommens auf dem entsprechenden Interface. Die maximale Verzögerung eines Datenpakets in einer Output-Queue ist durch deren Größe begrenzt. Wird bei einer vollen Queue ein weiteres Datenpaket für diese empfangen, so wird dieses verworfen.

Da das Verkehrsaufkommen im Internet stark zeitabhängig ist und nicht durch Management-Mechanismen kontrolliert wird, ist die Queueing-Verzögerung eine wesentliche Ursache für Laufzeitschwankungen der Datenpakete in IP-Netzen [Jia04]. Ebenso werden die meisten Paketverluste in den IP-Netzen durch Überläufe der Output-Queues auf den Netz-Interfaces verursacht. Je nach Interface-Typ können Leitungsgeschwindigkeiten zwischen einigen kBit/s bis mehreren GBit/s in Kommunikationssystemen betragen [Laa03]. Ebenso variieren je nach Hard- und Software-Architektur die Längen der

Warteschlangen im System. Aus diesem Grund kann kein allgemein gültiger Grenzwert für die Verzögerung t_{queue} genannt werden. Nach [Cla04] sind beispielsweise Warteschlangen in Endsystemen mindestens 32-64 kByte lang. Damit ergibt sich eine maximale Warteschlangen-Verzögerung bei einer Anbindung an das Internet über ISDN oder DSL in der Größenordnung von mehreren Sekunden.

2.5 Fehler bei Netzmessungen

Bei der Messung der Netzwerk-Performance und der Dienstgüte werden zwei Parameter betrachtet: die übermittelte Informationsmenge und die Zeitpunkte des Aussendens bzw. des Empfangs der Datenpakete. Die innerhalb einer Zeitspanne beobachtete Informationsmenge als Anzahl übertragener Bytes oder Datenpakete kann exakt bestimmt werden. Die Bestimmung eines Zeitpunktes für das Aussenden bzw. für den Empfang ist fehlerbehaftet. Bei Netzmessungen muss außerdem die Zeit an unterschiedlichen Orten bestimmt werden. Für den Vergleich der Zeit an zwei unterschiedlichen Orten wird eine Bezugszeit benötigt. Im Internet wird als Bezugszeit häufig die *Coordinated Universal Time (UTC)* [UTC] verwendet.

2.5.1 Begriffe in Verbindung mit der Uhrensynchronisation

Für eine Betrachtung der Fehler bei der Zeitmessung müssen die dafür verwendeten Begriffe geklärt werden. Der überwiegende Teil der Literatur zur Uhrensynchronisation von Rechnern ist in englischer Sprache veröffentlicht. Eine Systematisierung der Begriffe in Verbindung mit der Uhrengenauigkeit in englischer Sprache ist z.B. in [Pax98b] zu finden. Nachfolgend wird die Verwendung der Begriffe in dieser Arbeit festgelegt:

- **Auflösung *(engl. resolution):*** Als Auflösung einer digitalen Uhr wird die minimale Zeitdifferenz genannt, die eine Uhr darstellen kann. Diese ist nicht direkt mit der Länge der Anzeige verbunden. So wird z.B. bei älteren Unix-Systemen die Systemuhr bei jedem Aufruf der Systemzeit mindestens um 10 ms inkrementiert, obwohl Mikrosekunden angezeigt werden. In solchem Fall ist die Auflösung der Uhr gleich 10 ms.
- **Offset *(engl. offset):*** Als Offset wird die Differenz zwischen der angezeigten Zeit und der Referenz-Zeit (z.B. der UTC) genannt.
- **Schlupf *(engl. skew):*** Als Schlupf der Uhren wird die Differenz der Frequenz der Uhr gegenüber der Referenz-Uhr bezeichnet. Man kann den Schlupf als erste Ableitung des Offsets darstellen.

- **Drift *(engl. drift):*** Als Drift einer Uhr wird die zweite Ableitung des Offsets bzw. die erste Ableitung des Schlupfes bezeichnet.
- **Synchronisation:** Von einer Synchronisation bzw. einer **Uhrensynchronisation** wird in Bezug auf zwei Uhren gesprochen, wenn die Differenz des Offsets dieser Uhren einen bestimmten Wert nicht überschreitet. Wird z.B. angegeben, dass zwei Uhren auf 1 ms synchronisiert sind, bedeutet dies, dass durch bestimmte Maßnahmen sichergestellt ist, dass die Differenz der Offsets dieser Uhren zu jedem Zeitpunkt geringer als 1 ms ist. Bei einer qualitativen Bewertung der Synchronisation der Uhren wird auch der Begriff **Genauigkeit** verwendet.
- **Güte einer Uhr:** Als Güte wird der maximale Schlupf einer Systemuhr bzw. des Oszillators, bezogen auf die Soll-Frequenz, bezeichnet. Die Güte ist dimensionslos.

2.5.2 Fehlerquellen und Fehlerfortpflanzung bei der Zeitmessung

Bei der Durchführung einer Messung in einem Datennetz sind zwei rudimentäre Aufgaben zu lösen. Zum einen sind die Attribute eines Datenpakets festzustellen und zum anderen ist der Zeitpunkt für ein bestimmtes Ereignis in Verbindung mit diesem Datenpaket. Zu den betrachteten Attributen gehört in erster Linie die Paketlänge. Je nach Untersuchung können weitere Attribute erfasst werden.

Für die Fehlerabschätzung werden die Fehlerkomponenten nach ihren Ursachen klassifiziert. Für jede Komponente wird anschließend das Maximum abgeschätzt. Aus diesen Komponenten kann anschließend das Maximum für den Gesamtfehler berechnet werden. Die Komponenten des Messfehlers sind in Abbildung 2.5 dargestellt.

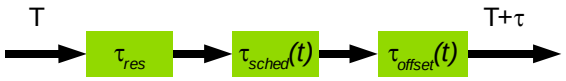

Abbildung 2.5: Blockschaltbild der Fehlerkomponenten bei der Zeitmessung

Die Systemzeit wird mit einer endlichen Auflösung fixiert, wodurch ein Auflösungsfehler τ_{res} entsteht.

Moderne Rechner werden als Mehrpozesssysteme realisiert, so dass die CPU-Zeit zwischen mehreren Prozessen geteilt wird. Somit kann es passieren, dass zwischen dem Eintreffen eines Ereignisses und dem Einholen der Systemzeit die CPU zwischenzeitlich

einem anderen Prozess zugewiesen wird. Die Zeitspanne zwischen der Zeit eines Ereignisses und der Zeit, an der die System-Uhr abgefragt worden ist, ist in Abbildung 2.5 als τ_{sched} bezeichnet. Je nach Definition des Messpunktes kann die Verweildauer eines Datenpakets in einem Buffer, bis es vom IP-Stack an die Anwendung gegeben wird, ebenfalls als Messfehler betrachtet (siehe Kapitel 5.1.4.2) werden. In diesem Fall ist er ebenfalls der Komponente τ_{sched} zuzuordnen, weil er ebenfalls durch das Process-Scheduling verursacht wird.

Die Uhr eines Systems hat einen Schlupf und eine Drift, verursacht durch Schwankungen der Frequenz der Oszillatoren sowie durch den Einfluss externer Ereignisse, wie z.B. durch die Korrektur der Frequenz vom NTP-Daemon [Mil96] . Damit ist der Offset der Systemuhr und dadurch auch der Fehler $\tau_{offset}(t)$ zeitabhängig. Außerdem kann der Offset einer Uhr durch externe Ereignisse – z.B. durch Synchronisationsmaßnahmen der Uhr (siehe Kapitel 3.2.5.1) – verändert werden.

Das Abfragen der Systemzeit nimmt in modernen Rechner-Systemen nur wenige Nanosekunden in Anspruch und wird somit nachfolgend vernachlässigt.

Die oben beschriebenen Komponenten des Messfehlers treten bei jeder Messung auf und können als eine Reihenschaltung von Fehlern dargestellt werden. Sind die einzelnen Komponenten zeitinvariant, so ist der maximale Gesamtfehler bei der Erfassung der Zeit nach [Mer77, S. 37] gleich der Summe der maximalen Fehlerbeträge einzelner Fehlerquellen. Bei einer Grenzwert-Betrachtung werden die zeitabhängigen Komponenten durch die Beträge der Maxima der Fehler ersetzt. Damit kann eine Berechnungsvorschrift für den maximalen Fehlerbetrag $\tau_{ges,max}$ bei der Messung der Zeit mit $\tau_{res,max}, \tau_{sched,max}, \tau_{offset,max}$ für die jeweiligen Maxima der Fehlerbeträge wie folgt dargestellt werden:

$$\tau_{ges,max} = \tau_{res,max} + \tau_{sched,max} + \tau_{offset,max} \tag{2.12}$$

2.6 Anforderungen an Messsysteme

In Kapitel 1 wurde bereits dargestellt, dass Netzmessungen die einzige Möglichkeit darstellen, das Verhalten von Datenströmen in IP-Netzen abzuschätzen oder vorherzusagen. Nachfolgend werden die Anforderungen vorgestellt, welche bei der Wahl bzw. bei der Entwicklung eines Messsystems zu berücksichtigen sind. Anschließend wird in Kapitel 3 eine Auswahl der Hard- und Software getroffen sowie ein geeigneter Ansatz für Messungen in IP-Netzen erarbeitet.

2.6.1 Verfügbarkeit der verwendeten Plattform

Eine wesentliche Rolle bei der Wahl eines Messsystems spielt die Verfügbarkeit der verwendeten Hardware- und Softwareplattform. In Anbetracht der Vielfalt der ver-

wendeten Netzwerktechnologien kommt ein Vorhalten von unterschiedlichen Hardware-Werkzeugen für Messungen in unterschiedlichen Netzen in vielen Fällen aus wirtschaftlichen Gründen nicht in Betracht. Bei einem schnellen Technologiewechsel sind bei hardwarebasierten Lösungen häufig teure Neuanschaffungen erforderlich. Speziell bei der Anpassung an neue Netzwerktechnologien haben softwarebasierte Lösungen den Vorteil eines einfachen und in der Regel kostengünstigen Updates. In den letzten Jahren findet die günstige IBM-PC-Hardware-Architektur zunehmend Verbreitung. Für diese Hardware existiert eine große Auswahl an Netzwerk-Interface-Karten für unterschiedliche Netz-Technologien. In Verbindung mit Software-Lösungen können damit kostengünstige Messumgebungen für IP-Netze auf unterschiedlichen Medien realisiert werden. Eine kostengünstige Messumgebung wird unter anderem in [Ubi01] vorgestellt.

Bei einer softwarebasierten Lösung stellt sich ebenso die Frage der Wahl eines geeigneten Betriebssystems. Im Bereich der Netzwerk-Untersuchungen sowie des Netzwerk-Managements finden zunehmend PC-Systeme mit quelloffenen Betriebssystemen, wie z.B. Linux oder BSD-Unix Anwendung. Des Weiteren sind im Bereich des Netzwerk-Managements SPARC/Solaris-basierte Systeme weit verbreitet. Im Desktop-Bereich sind Windows-Systeme von Microsoft am weitesten verbreitet.

Wird ein Messsystem unter dem Gesichtspunkt der Portierbarkeit auf unterschiedliche Systeme entwickelt, so ist dies mit relativ geringem Aufwand zwischen unterschiedlichen Unix-Derivaten realisierbar. Ebenso ist eine portable Implementierung zwischen unterschiedlichen Windows-Varianten einfach realisierbar. Eine Portierung von Hochperformance-Messsystemen zwischen Unix und Windows ist aber wegen unterschiedlicher Interfaces zu den hochaufgelösten Uhren des Betriebssystems nur mit einem größeren Aufwand machbar. Im Bereich der Rechenzentren sowie der Internet Service Provider wird häufig unter den Gesichtspunkten der Möglichkeit des entfernten Zugriffs sowie der Verfügbarkeit einer komfortablen Shell Unix-basierten Messsystemen der Vorzug gegeben.

2.6.2 Flexibilität

Ein Messsystem soll dem Nutzer einen hohen Grad an Flexibilität bieten. Bei dem Entwurf eines Systems für das Messen im Netz ist es von Vorteil, wenn mit diesem ein breites Spektrum an Messaufgaben realisiert werden kann.

Die Forderung nach Flexibilität tangiert auf der einen Seite die Problematik der Vergleichbarkeit der Ergebnisse. So ist es speziell für Größen, für die keine Metriken definiert sind, mit einem Messtool einfacher, vergleichbare Ergebnisse zu produzieren als bei Verwendung unterschiedlicher Tools. Außerdem betrifft diese Forderung das nachfolgend diskutierte Thema der Bedienung des Programms. Messungen in modernen Datennetzen erfordern die Einstellung einer Vielzahl von Parametern, was eine nicht unerhebliche Einarbeitungszeit in die Bedienung eines Messtools erfordert. Wird statt

mehrerer Messsysteme nur ein Tool für unterschiedliche Messungen verwendet, kann dies eine wesentliche Ersparnis der Einarbeitungszeit bedeuten.

2.6.3 Konfiguration und Bedienung

Einen wesentlichen Aspekt der Nutzung eines Messtools bilden die Konfiguration und die Bedienung des Systems. Dabei muss grundsätzlich zwischen Systemen mit einem grafischen User-Interface (GUI) und Systemen mit einem Kommandozeilen-Interface unterschieden werden.

Systeme mit einem grafischen User-Interface haben den Vorteil einer intuitiven Bedienung. Dadurch ist eine schnellere Einarbeitung in die Bedienung gegeben als bei Systemen mit einem Kommandozeilen-Interface. Eine effiziente Fernsteuerung GUI-basierter Anwendungen ist aber in der Regel nicht einfach realisierbar.

Systeme mit einem Kommandozeilen-Interface werden in einer Shell gestartet. Auf Unix-Systemen sind solche Programme stark verbreitet. Das Programm wird dabei durch die Angabe von Kommandozeilenparametern oder über Konfigurationsdateien gesteuert. Zu den wesentlichen Vorteilen solcher Systeme gehört die Möglichkeit eines effizienten Fernzugriffs, zum Beispiel über eine Secure Shell [Ylo04]. Die Software lässt sich in andere Systeme einbetten. Damit sind unter anderem Web-basierte Messsysteme realisierbar. Ein Nachteil der Kommandozeilen-Interfaces ist die weniger intuitive Bedienung als bei GUI-basierten Anwendungen. Um diesen Nachteil zu eliminieren, Interfaces nachgerüstet werden, z.B. mit Hilfe der CGI-Schnittstelle [Rob95]. Dabei können die grafischen Interfaces einfach an die Anforderungen konkreter Projekte angepasst werden.

Netzmessungen werden häufig als Messreihen mit veränderlichen Parametern durchgeführt. Dabei werden für umfangreichere Messreihen Scripte (z.B. Perl- oder Shell-Scripte [Kri01]) mit variablen Parametern gestartet. Hierfür sind wiederum Kommandozeilen-Programme besser geeignet.

2.6.4 Auswertung der Messdaten

Es ist vorteilhaft, wenn einige Größen, wie z.B. die Anzahl verloren gegangener Datenpakete, die mittleren, minimalen oder maximalen Laufzeiten der Datenpakete direkt nach Abschluss der Messung ausgegeben werden können. Komplexere statistische Auswertungen können zu einem späteren Zeitpunkt durchgeführt werden. Dafür ist es erforderlich, auf die Ergebnisse einzelner Messpunkte zurückzugreifen.

In [Vay04] wird gezeigt, dass für die Auswertung großer Datenmengen eine grafische Darstellung besser geeignet ist als eine textuelle oder tabellarische Form. Folglich wer-

den weitere Werkzeuge benötigt, mit welchen einschlägige statistische Auswertungen, wie z.B. die Häufigkeitsverteilung der Werte, CDF und CCDF, in einer geeigneten Form erzeugt werden können.

2.7 Existierende Tools für die Messung der Netzwerk-Performance

Für die Untersuchung der Netzwerk-Performance in IP-Netzen existieren bereits mehrere Tools. Nachfolgend werden häufig verwendete Messsysteme kurz vorgestellt. Eine ausführliche Übersicht existierender Tools ist in [Zse02] zu finden.

Das älteste und bekannteste Tool zum Messen der Netzwerk-Performance ist das Programm *ping*. Für die Messung werden dabei *Echo-Requests* über das *Internet Control Message Protocol (ICMP)* [Pos81b] an den entfernten Host gesendet. Empfängt das Endsystem einen Echo-Request, so wird dieser sofort kopiert und an den Absender zurückgeschickt. Somit können mit *ping* die Umlauf-Verzögerung und der Umlauf-Paketverlust sowie der Jitter gemessen werden. Weitere Details zu der Funktionsweise von *ping* sind in [Ste94a, Kap. 7] zu finden. Zu den Vorteilen des Programms gehört seine weite Verbreitung. Das Programm *ping* ist in nahezu allen Betriebssystem vorzufinden, welches TCP/IP unterstützt. Außerdem ist die Server-Komponente von *ping* im Kernel der Betriebssysteme realisiert. Zu den Nachteilen gehört die geringe Flexibilität und die Ungenauigkeit der Messergebnisse. Außerdem sind die Messergebnisse nicht immer repräsentativ, da ICMP-Pakete in einigen Netzknoten anders behandelt werden als TCP- oder UDP-Datenpakete.

Ein anderes, weit verbreitetes Tool ist *traceroute*. Mit *traceroute* kann der Pfad zwischen zwei Endpunkten im Netz bestimmt werden. Details zu diesem Tool sind in [Ste94a, Kap. 8] zu finden. Es ist ebenso wie *ping* in nahezu jedem Betriebssystem implementiert, welches TCP/IP-unterstützt. Der wesentliche Nachteil von *traceroute* ist, dass die von dem Programm ausgesendeten bzw. empfangenen Nachrichten häufig von Routern und Firewalls blockiert werden und somit keine Konnektivität geprüft werden kann. Die Nachteile von *ping* und *traceroute* soll das von der IPPM Working Group zurzeit entwickelte OWAMP-Protokoll beseitigen (siehe Kapitel 2.1.2.1). Es stehen aber noch Implementierungen des Protokolls aus. Das Protokoll adressiert in erster Linie die Problematik der Performance-Messung zwischen IP-Routern und wird deshalb in dieser Arbeit bei der Betrachtung der Ende-zu-Ende-Messungen nicht weiter berücksichtigt.

Ein weiteres Tool für die Messung der Netzwerk-Performance ist *TTCP* [Muu00]. Das Programm wurde 1984 für das BSD-Unix entwickelt. Später ist es weiterentwickelt und erweitert sowie auf unterschiedliche Unix-Derivate und auf Windows portiert worden. Das Haupt-Einsatzgebiet des Tools ist die Messung des Durchsatzes über TCP- und UDP. Zu den Vorteilen des Programms gehört seine weite Verbreitung. Ende

2002 wurde *TTCP* um die Fähigkeit erweitert, mehrere *TTCP*-Sender mit einem Empfänger simultan zu verbinden. Es wird aber keinerlei Synchronisation zwischen einzelnen Empfänger-Instanzen vorgenommen. Damit können die Messergebnisse durch die gegenseitige Störung unterschiedlicher Datenströme verfälscht werden. Außerdem besteht nur sehr eingeschränkt die Möglichkeit, Steuerinformationen zwischen Sender und Empfänger auszutauschen. Die Einsatzmöglichkeit von *TTCP* ist im Wesentlichen auf das Messen des Durchsatzes beschränkt.

Das Tool *Iperf* [Tir04, Tir03] wird von dem National Laboratory for Applied Network Research, USA (NLANR), entwickelt und wird von den Autoren als Nachfolger von TTCP angesehen. Ebenso wie TTCP kann der Durchsatz mit Hilfe von TCP- und UDP-Paketen gemessen werden. Die Datenquelle wird als eine Constant-Bitrate-Quelle realisiert. Ein simultanes Messen mit mehreren Datenströmen ist möglich. Dabei steht die gegenseitige Störung von Datenströmen untereinander im Vordergrund der Untersuchungen [Hac02]. Es ist mit *Iperf* z.B. nicht möglich, Jitter und Verzögerungen einzelner Datenpakete zu messen.

Bei keinem der oben vorgestellten Tools steht ein GUI zur Verfügung.

Eine Entwicklergruppe der Tampere University of Technology in Finnland hat einen abweichenden Ansatz bei der Entwicklung des Messtools *Rude/Crude* verfolgt [Ubi01]. Mit Rude/Crude wird in einem Einweg-Szenario (siehe Kapitel 3.2.5) gemessen. Mit diesem System können Paketverluste, Einweg-Verzögerungen sowie Jitter gemessen werden. Die Verwendung des Systems bei Netzmessungen wird in [Ubi01, Smo03] vorgestellt. Beim Aussenden der Datenpakete können unterschiedliche Datenquellen emuliert werden.

Das System hat aber eine Reihe von Einschränkungen. So wird nur die Messung mit UDP-Paketen unterstützt. Es können maximal 100.000 Samples ausgewertet werden. Mit unterschiedlichen Quellenmodellen können nur Kurzzeit-Messungen realisiert werden.

Das System wird zurzeit von den ursprünglichen Autoren nicht weiterentwickelt. Der Programmcode wird von der Internet-Community [RUD04] gepflegt. Das System ist primär für das Betriebssystem Linux entwickelt worden. Es lässt sich aber auch unter Solaris übersetzen. Von der Internet-Community wurde eine abgewandelte Variante mit einer GUI namens *Grude* Ende 2002 entwickelt.

Brix Verifier [Bri04] ist eine Reihe von Hardware-Lösungen zur QoS-Messung, welche vom Unternehmen Brix Networks entwickelt werden. Mit den *Brix Verifiern* werden aktive und passive Messungen im Netz durchgeführt. Nach Angaben des Herstellers soll die Güte von einzelnen Netz-Anwendungen, wie z.B. VoIP, SMTP, NNTP, gemessen werden. Details zur Reailsierung von Quellenmodellen dieser Anwendungen sind öffentlich nicht verfügbar. Der Fehler der Messung der Einweg-Verzögerung liegt in der Größenordnung von etwas unter einer Millisekunde und ist damit in etwa ebenso groß

wie der erwartete Fehler bei der Durchführung von Einweg-Messungen in Software auf Unix-basierten Systemen (siehe Kapitel 3.2.5.2). Es stehen Schnittstellen für 10, 100, und 1.000 MBit/s Ethernet-Netze sowie für ATM-Netze bis 155 MBit/s zur Verfügung. Die Bedienung der Systeme erfolgt über eine grafische Oberfläche.

Hardwarebasierte hochpräzise passive Messungen der Netzwerk-Performance sind mit Systemen des Unternehmens Acterna [Act04] möglich. Rudimentäre aktive Messungen – mit einer CBR-Quelle zwischen zwei Punkten im Netz sind ebenfalls möglich.

Die vorgestellte Übersicht der Tools zur Messung der Netzwerk-Performance zeigt die wesentlichen Richtungen sowie den Stand der Entwicklungen in diesem Bereich. Es wurden außerdem einige Messsysteme eng auf ein Projekt oder auf eine Arbeit zugeschnitten und nach Abschluss entsprechender Untersuchungen nicht weiterentwickelt. So waren im Rahmen von Projekten am Lehrgebiet Rechnernetze und Verteilte Systeme der Universität Hannover zwei Tools zur Messung der Umlauf-Verzögerung namens *RTest* [Grü98] und *TTest* entwickelt worden. Beide Systeme werden aber nicht weiter gepflegt. Außerdem werden viele Systeme für spezielle Untersuchungen entwickelt, wie z.B. die Tools *tcpanaly* [Pax97] und *Web100* [Web04], mit welchen einzelne Untersuchungen der Netzwerk-Performance untersucht werden – z.B. die Link-Datenrate der langsamsten Strecke zwischen zwei Punkten im Netz sowie die Effizienz der TCP-Steuerungsmechanismen.

Trotz der Vielfalt der verfügbaren Tools zur Messung der Netzwerk-Performance bzw. der Dienstgüte genügen diese den aktuellen Anforderungen bei den Untersuchungen in IP-Netzen nicht. Aus den vorgestellten Systemen ist nur *Rude/Crude* sowie *Iperf* für aktive Einweg-Messungen geeignet. Eine Nachbildung bestimmter Anwendungen ist nur mit *Rude/Crude* möglich. Diese Möglichkeit zur Nachbildung von Anwendungen ist jedoch sehr eingeschränkt.

Keines der vorgestellten softwarebasierten Systeme ist für eine verteilte Messung (siehe Kapitel 3.2.6) geeignet. Es fehlt an systematischer Analyse der Messfehler bei den existierenden Tools für aktive Einweg-Messungen.

2.8 Projekte und Initiativen im Bereich Netzmessungen

Nachfolgend werden einige Projekte und Initiativen vorgestellt, die sich mit dem Thema Messung der Netzwerk-Performance bzw. mit eng verwandten Themen beschäftigen.

- **Quasar**
  Ziel des vom DFN geförderten Projektes Quasar [Qua03] ist es, eine Quality-of-Service-Architektur für das G-WiN aufzubauen. Im Rahmen des Projektes werden unter anderem Ansätze für passive und aktive Messungen im G-WiN bewertet.

Dabei soll die Wirksamkeit unterschiedlicher QoS-Mechanismen analysiert werden. Mit Hilfe von passiven Messungen wird unter anderem der Ressourcenbedarf von unterschiedlichen Anwendungen im IP-Netz untersucht. An dem Projekt sind das Fraunhofer-Institut für offene Kommunikationssysteme (FOKUS), das Rechenzentrum der Universität Stuttgart (RUS) sowie das Institut für Nachrichtenvermittlung und Datenverarbeitung der Universität Stuttgart beteiligt.

- **Sequin**
  Das von der Europäischen Kommission geförderte Projekt „Service Quality across Independently Managed Networks“ (SEQUIN) [SEQ04] beschäftigt sich mit der Entwicklung eines einheitlichen Dienstgüte-Überwachung- und Management-Systems, verteilt über mehrere nationale Forschungsnetze. Hierbei spielen die Definitionen sowie die Überwachung der Service Level Agreements eine wichtige Rolle. Einige Aspekte aktiver und passiver Messungen werden hier ebenfalls behandelt.

- **CAIDA**
  Die Organisation *„Cooperative Association for Internet Data Analysis“ (CAIDA)* hat das Ziel, die Entwicklung von Systemen zur Analyse von Datenströmen im Internet zu koordinieren [CAI04]. Dabei wird von CAIDA unter anderem die Entwicklung geeigneter Tools für die Durchführung von Netzmessungen gefördert und koordiniert. Das oben vorgestellte Messtool *Iperf* ist im Rahmen der CAIDA-Aktivitäten entstanden. Des Weiteren werden von CAIDA Umlauf-Verzögerungen zwischen mehreren Punkten in Internet über längere Zeiträume gemessen und die Ergebnisse für Forschungsprojekte zur Verfügung gestellt. Zu den von CAIDA koordinierten Projekten gehören unter anderem:

  – **Bandwidth Estimation Project (BW-EST)**. Das Projekt behandelt die Anforderungen an Tools für das Messen der Dienstgüte von Grid-Anwendungen.
  – **Analysis and Visualization of IP Connectivity**. Hier werden Themen der Erfassung und Darstellung der Güte von Verbindungen zwischen unterschiedlichen Providern im Internet behandelt.

- **RIPE NCC**
  Im Rahmen der Initiative *„Test Traffic Measurements“ des „Réseaux IP Européens Network Coordination Centre“ (RIPE NCC)* [RIPE] wurde ein Messsystem entwickelt, das mittlerweile bei mehreren Internet Service Providern als so genannte „*RIPE-Box*“ zum Einsatz kommt. Diese besteht aus einer speziell eingerichteten PC-Hardware mit einer GPS-synchronisierten Uhr (siehe Kapitel 3.2.5.1). Aktive Einweg-Messungen werden von einer vorinstallierten und vorkonfigurierten Software durchgeführt. Die benutzerdefinierte Konfiguration sowie

das Abrufen von Messergebnissen erfolgt dabei über eine Web-Schnittstelle. Eine Beschreibung des Systems ist in [Geo01] gegeben.

- **Netzwerk Monitoring im G-Win**
  Seit dem Frühjahr 2004 werden zwischen einzelnen Kernnetzknoten des G-WiN aktive Einweg-Messungen vom DFN durchgeführt. Die Ergebnisse dieser Messungen können im Internet über das Web-Interface dargestellt werden [Kle03]. Dieser Service basiert auf den Entwicklungen des WiN-Labors in Erlangen [Win04].

- **Grid**
  In der *Network Measurements Working Group – (NMWG)* [NMWG04] des *Global Grid Forums* [GGF04] werden Aspekte der Netzmessungen und der Realisierung von Quality-of-Service-Mechanismen in Verbindung mit dem Grid-Computing behandelt. Eines der Ziele der Arbeitsgruppe ist die Evaluierung von Messtools, die für das Grid-Computing verwendbar sind.

Kapitel 3

Messszenarien und Problemanalyse

3.1 Messen aus Sicht der Nutzer und Anbieter

Die Diskussion der Grundlagen zur Quality of Service in Kapitel 2 zeigt auf die Vielfältigkeit des Problems der Messung der Netzwerk-Performance bzw. der Dienstgüte. In Kapitel 2.1.4 werden drei Rollen dargestellt, welche die Anbieter und die Nutzer von Diensten im Internet annehmen können. Je nach angenommener Rolle ergeben sich unterschiedliche Sichten auf das Messen im Netz (siehe Abbildungen 3.1 bis 3.3).

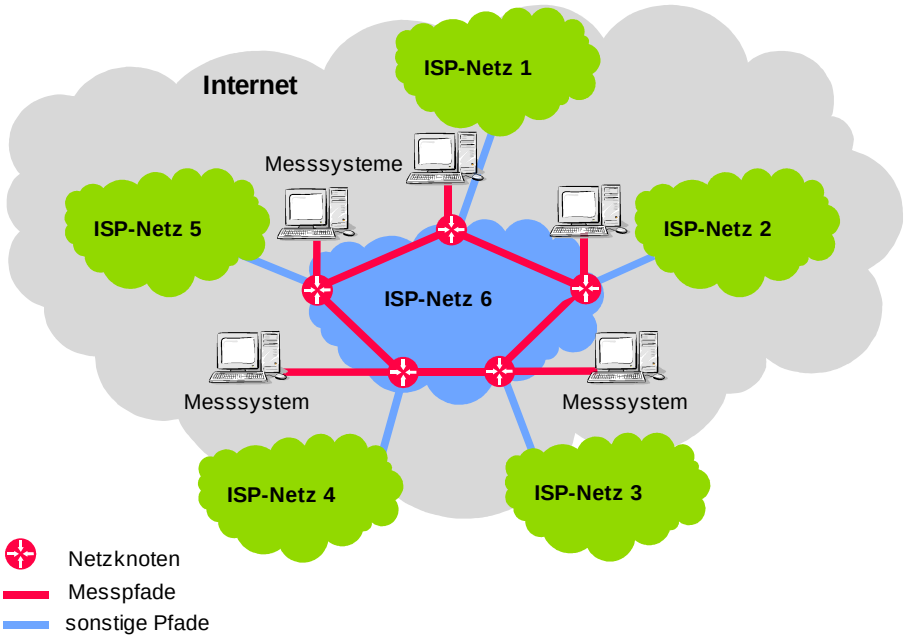

Abbildung 3.1: Messszenario aus der Sicht eines Netzbetreibers (ISP)

1. Für Netzbetreiber (ISP) ist in erster Linie die Netzwerk-Performance zwischen unterschiedlichen Übergängen in die Netze anderer Provider sowie im inneren

Netz von Interesse (siehe Abbildung 3.1). Die Auswertung von Messergebnissen wird dabei zur Dimensionierung und Netzplanung herangezogen. Ebenso kann eine dauerhafte Überwachung der Einhaltung von SLAs mit Hilfe solcher Messungen von Interesse sein. Bei der Überwachung der Netzwerk-Performance werden zunehmend Charakteristiken bzw. spezielle QoS-Anforderungen einzelner Anwendungen berücksichtigt.

2. Für Dienstanbieter (ASP) steht im Mittelpunkt der Untersuchungen die Erreichbarkeit eigener Server aus den Netzen unterschiedlicher ISPs. Zu diesem Zweck wird die Dienstgüte von unterschiedlichen Netzen im Internet zu eigenen Servern untersucht. Ein typisches Messszenario für derartige Untersuchungen ist in Abbildung 3.2 dargestellt. Dabei wird im lokalen Netz, in dem die Server angeschlossen sind, ein Mess-Server aufgestellt, gegen welchen Messungen durchgeführt werden können. Entfernte Messsysteme werden in der Nähe typischer Lokationen der Nutzer platziert.

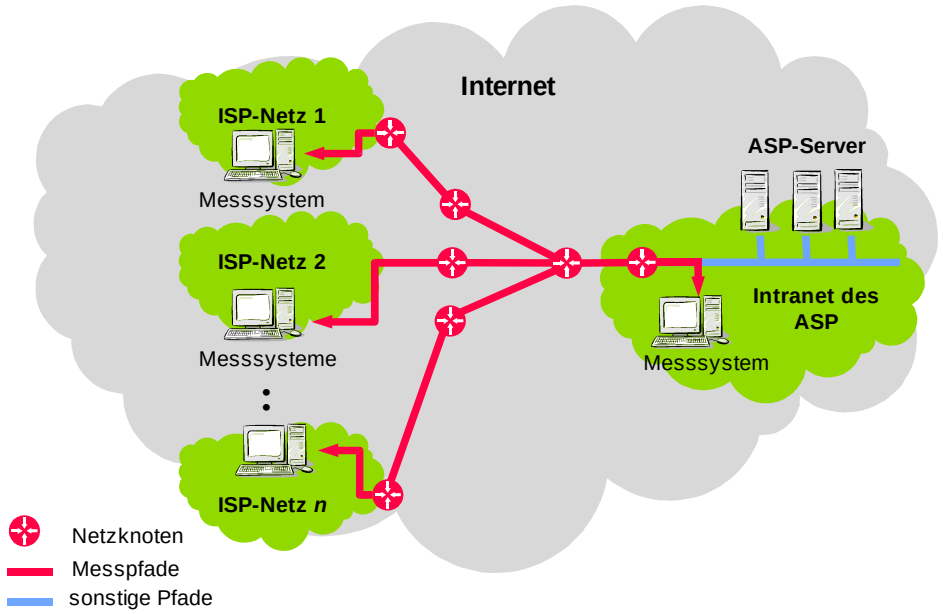

Abbildung 3.2: Messszenario aus der Sicht eines Dienstanbieters (ASP)

3. Ein Messszenario für die Messung der Dienstgüte seitens der Nutzer der IP-Dienste ist in Abbildung 3.3 dargestellt. Dabei konzentriert sich die Betrachtung

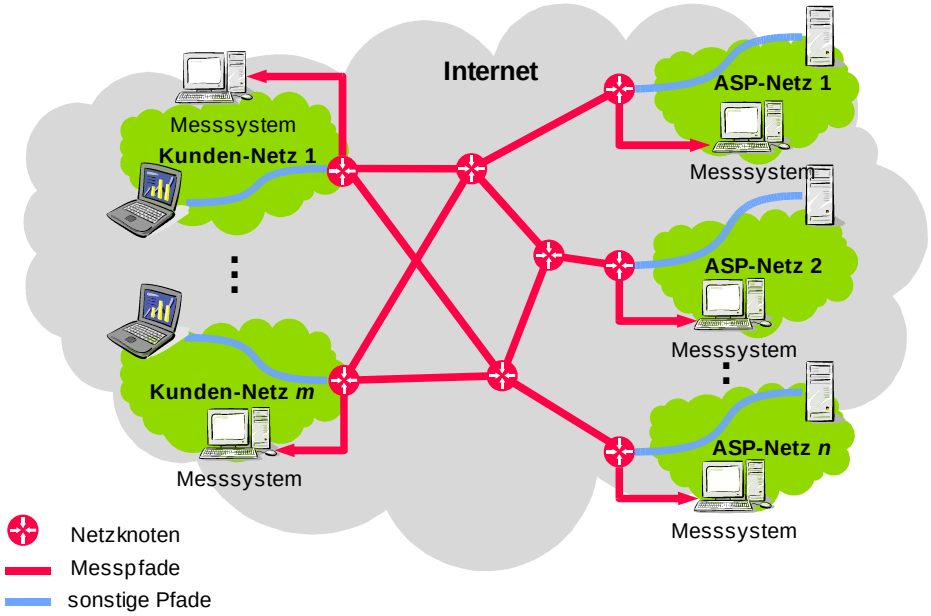

Abbildung 3.3: Messszenario aus der Sicht der Nutzer des IP-Netzes

auf die Qualität der Verbindung vom Anschluss des Teilnehmers zu unterschiedlichen Servern, von welchen Dienste in Anspruch genommen werden. Die gewonnenen Erkenntnisse können unter anderem Grundlage für eine Entscheidung sein, von welchem Server ein Dienst angefordert werden soll. Der letzte Punkt gewinnt speziell bei Grid- und Peer-to-Peer-Anwendungen zunehmend an Bedeutung. Dabei wird anhand der erwarteten Dienstgüte entschieden, von welchem Ort ein File, ein Video oder sonstige Information abgerufen wird.

Den drei Sichten auf das Messen sind folgende Aspekte gemeinsam:

- Speziell im Bereich industrieller Kunden bilden zunehmend Service Level Agreements die Basis für die Beziehung zwischen Netz- bzw. Dienstanbietern und den Kunden. Dies erfordert eine gute Kenntnis der Güte der IP-Netze sowohl seitens der Anbieter als auch der Kunden.
- Eine einfache Punkt-zu-Punkt-Messung ist in den Szenarien nicht ausreichend, da die Parameter von mehreren und zu mehreren Punkten zu bestimmen sind. Es ist im Allgemeinen eine Durchführung von *n:m*-Messungen zwischen unterschiedlichen Punkten im Netz notwendig (siehe Kapitel 3.2.6).

- Je nachdem welche Anwendungen im Netz betrachtet werden, müssen die Messungen unterschiedlich gestaltet werden. Stehen z.B. VoIP-Anwendungen im Vordergrund, so ist zu untersuchen, wie das Netz UDP-Pakete geringer Länge vermittelt. Werden hingegen Grid-Anwendungen betrachtet, so muss das Netz für einen sehr hohen Durchsatz für TCP-Verkehr ausgelegt sein. Die beiden dargestellten Gesichtspunkte sind mit einer einzigen Messung nicht abzudecken.

3.2 Erarbeitung des Messszenarios

Eins der Ziele der Arbeit ist die Entwicklung eines Messsystems, welches ein breites Spektrum von Messaufgaben abdeckt. Dieses soll ohne Modifikation des Betriebssystems auf einer Standard-Hardware und -Software lauffähig sein. Es sollen verteilte Messungen der Dienstgüte möglich sein. Dabei sollen außerdem Parameter einzelner Anwendungen in der Messung abgebildet werden.

3.2.1 Wahl der Hardware- und Software-Plattform

Aus Gründen der Flexibilität (siehe Kapitel 2.6) fällt die Wahl auf eine softwarebasierte Lösung. Dafür müssen eine Hardware-Plattform und ein Betriebssystem ausgewählt werden, für welche das Messsystem entwickelt werden soll. Anschließend werden ebenfalls auf Basis der Überlegungen in Kapitel 2.6 die sonstigen Anforderungen an das zu entwickelnde Messsystem erarbeitet.

Die Wahl einer geeigneten Hard- und Software wird unter den Gesichtspunkten der Verfügbarkeit der Plattform und der Kosten getroffen. Damit kommen in erster Linie IBM-kompatible PCs in Betracht. Moderne PCs bieten ein breites Spektrum an Netzwerk-Interfaces. Mit Prozessor-Taktfrequenzen oberhalb der 1-GHz-Grenze sind prinzipiell Messungen im Bereich unterhalb einer Mikrosekunde möglich. Es ist aber von Vorteil, wenn das System neben der PC-Architektur auf weiteren Hardware-Plattformen lauffähig ist.

Als Betriebssysteme für PC-Systeme kommen aus Gründen der Verfügbarkeit der Software Microsoft Windows sowie Unix-Derivate, wie z.B. Linux und BSD, in Betracht. Wegen der Inkompatibilitäten der APIs ist zwischen einer Windows- und einer Unix-Implementierung zu wählen. Diese Wahl wird durch folgende Faktoren bestimmt:

- Um eine hohe Präzision der Messung bei hohen Datenraten zu erreichen, sind ggf. administrative Eingriffe in das Rechnersystem notwendig. Es müssen beispielsweise alle nicht zwingend erforderlichen Dienste und Prozesse abgeschaltet sowie Prioritäten von Hintergrund-Prozessen geändert werden. Dank der Verwendung

gemeinsamer Konzepte des System V [Bou88] sowie des BSD-Systems [Kus90] im Aufbau und in der Verwaltung aktueller Unix-Implementierungen bieten Unix-Betriebssysteme mehr Transparenz und Übersichtlichkeit bei der Verwaltung als Windows-Betriebssysteme.

- Für Untersuchungen mit besonders hohen Anforderungen an die Güte der Messungen kann es beispielsweise notwendig sein, die grafische Oberfläche des Betriebssystems zu deaktivieren, um damit die Hintergrund-Last und die Interrupt-Rate zu reduzieren. Bei aktuellen Windows-Systemen ist dies nicht möglich.
- Unix-Workstations sowie Server-Systeme haben in Rechenzentren und bei ISPs eine starke Verbreitung. Somit ist es von Vorteil, wenn das System auf solchen Systemen lauffähig ist.
- Für Unix-Systeme steht eine größere Auswahl an Prozessoren und Hardware-Architekturen zur Verfügung als für Windows-Systeme. Dieser Faktor spielt insbesondere für zukünftige Erweiterungen und für den Einsatz des Systems in Forschungsarbeiten eine wesentliche Rolle.
- Über die Unix-Shell-Schnittstelle ist eine effiziente Fernsteuerung sowie eine Anbindung an einen Web-Server sowie an Netzmanagement-Systeme möglich. Solche Anbindung ist für eine zukünftige Realisierung von Diensten zur Überwachung der QoS und der Netzwerk-Performance notwendig. Bemerkenswert ist weiterhin die Tatsache, dass alle in Kapitel 2.7 vorgestellten softwarebasierten Tools über die Kommandozeile bedient werden und ursprünglich für Unix entwickelt worden sind.

Aus den vorgestellten Überlegungen ist eine Implementierung auf Unix-Systemen zu bevorzugen. Für die Arbeit standen Rechner mit Intel-Pentium Prozessoren unter Linux in den Kernel-Versionen 2.4 und 2.6 sowie SPARC-Systeme mit dem Betriebssystem Solaris in den Versionen 8 und 9 zur Verfügung. Die grundsätzliche Funktion wurde auf diesen Systemen getestet. Genauere Untersuchungen sind auf Linux-Systemen durchgeführt worden.

3.2.2 Prozess-Scheduling im Betriebssystem

Die im vorhergehenden Abschnitt genannten Betriebssysteme sind interaktive Mehrprozesssysteme. Dabei ist das Prozess- bzw. Thread-Scheduling für die Verwaltung der Prozessor-Ressourcen zuständig [Sil98, S. 123]. Bei der Realisierung zeitkritischer Anwendungen, zu welchen das zu entwickelnde Messsystem zählt, ist die *Antwortzeit* eines Prozesses maßgebend für die Güte des Messsystems. Die *Antwortzeit* ist die Zeit, die

ein Prozess auf die Zuteilung der CPU warten muss, wenn er lauffähig ist und die CPU von einem anderen Prozess belegt ist. Diese Zeit hängt vom Scheduling-Verfahren des Betriebssystems ab sowie von dessen Parametern, wie z.B. dem Quantum (siehe unten).

Bei Mehrprozesssystemen wird zwischen *interaktiven* und *Echtzeit-Scheduling-Verfahren* unterschieden [Tan01, Kap. 2.5]. Bei den ersten hängt die Antwortzeit eines Prozesses von Faktoren ab wie die verbrauchte CPU-Zeit und die Anzahl konkurrierender Prozesse im System. Es kann kein Maximum für die Antwortzeit angegeben werden. Bei Echtzeit-Scheduling-Verfahren wird zwischen einer *harten* und *weichen Echtzeitfähigkeit* unterschieden. Bei einer harten Echtzeitfähigkeit werden bestimmte Grenzwerte der Antwortzeiten deterministisch eingehalten. Bei der weichen Echtzeitfähigkeit wird mit einer großen Wahrscheinlichkeit die vorgegebene maximale Antwortzeit eines Prozesses nicht überschritten.

Viele Anwendungen, welche eine weiche Echtzeitfähigkeit voraussetzen, werden gegenwärtig auf Standard-Unix- sowie Windows-Systemen erfolgreich realisiert. Dazu gehören beispielsweise Fax- sowie Sprach- und Video-Anwendungen. Für das Erreichen einer weichen Echtzeitfähigkeit stehen in Unix-Betriebssystemen das Round-Robin und das FIFO-Scheduling-Verfahren zur Verfügung [Set02]. Dabei werden diese Echtzeit-Scheduling-Verfahren mit interaktiven Verfahren nebenläufig angewendet.

Die maximale Antwortzeit eines Prozesses wird unter anderem durch die zeitliche Auflösung der Arbeit des Schedulers bestimmt. Moderne Betriebssysteme wie Solaris, Linux, BSD und Windows arbeiten standardmäßig mit einer Auflösung des Schedulers von 10 ms [Ets03]. Dabei kann ein Prozess je nach Implementierung einen oder mehrere Zeitschritte auf der CPU ununterbrochen verweilen. Die minimale CPU-Zeit, die das Betriebssystem einem Prozess zuweisen kann, wird als *Quantum* bezeichnet. Ist eine Prozesspriorisierung im System möglich, so ist die maximale Wartezeit eines am höchsten priorisierten Prozesses gleich dem Quantum. Somit beträgt die maximale Wartezeit des am höchten priorisierten Prozesses auf den in Betracht kommenden Betriebssystemen mindestens 10 ms. Dies ist für eine gute Approximation einer Echtzeit-Fähigkeit im Mikrosekunden-Bereich nicht ausreichend. Die Auswirkung der zeitlichen Auflösung des Schedulers auf die Güte zeitkritischer Anwendungen wird ebenfalls in [Ets03] vorgestellt. Es ist zu beachten, dass die Verwendung von Echtzeit-Scheduling-Verfahren in einem Programm eine Betrachtung des gesamten Rechnersystems erfordert. Es kann beispielsweise nicht sichergestellt werden, dass die oben genannte maximale Antwortzeit nicht überschritten wird, sobald mehrere Prozesse im System mit der höchsten Priorität versehen werden.

Für die Realisierung von zeitkritischen Anwendungen wird häufig *RTLinux* verwendet – eine Abwandlung von Linux mit einer harten Echtzeitfähigkeit [Yod99]. RTLinux kann auf Standard-Hardware verwendet werden. Die TCP/IP-Unterstützung für RTLinux ist

aber nicht bzw. nur teilweise gegeben [Pas03]. Somit kann es für die Realisierung des Messsystems nicht verwendet werden.

Die Implementierung des Messsystems auf einem Betriebssystem mit harter Echtzeitfähigkeit würde die Zielsetzung der Arbeit – ein Messsystem zu schaffen, welches auf weit verbreiteter Hardware und Software ohne Eingriffe in den Kernel lauffähig ist – nicht treffen.

3.2.3 Wahl zwischen aktiven und passiven Messungen

Für die Realisierung eines Messsystems ist eine Entscheidung zwischen passiven und aktiven Messungen zu treffen.

Bei passiven Messungen werden keine Datenpakete in das Netz emittiert. Vielmehr werden an bestimmten Aufpunkten im Netz Datenpakete gesammelt und analysiert. Damit wird der Datenverkehr nicht gestört. Die Durchführung von passiven Messungen erfordert aber den Zugang zu den Netzknoten, an welchen die betrachteten Datenströme vermittelt werden. In einem passiven Messszenario kann damit die Behandlung von Datenströmen beobachtet werden. Es ist aber nicht möglich, das Verhalten von Datenströmen mit konkreten Charakteristiken abzuschätzen, wenn diese im Netz nicht gegenwärtig präsent sind. Es ist im Allgemeinen keine Bestimmung von Einweg- oder Umlauf-Verzögerungen in einer passiven Messung möglich.

In einem aktiven Messszenario werden dagegen Datenpakete zwischen zwei Endpunkten im Netz ausgetauscht. Dafür ist lediglich ein Zugang zu zwei Punkten im Netz erforderlich, an denen die Messsysteme angeschlossen werden. Bei Umlauf-Messungen, z.B. mit dem Programm *ping*, ist der Zugang lediglich zu einem Punkt im Netz erforderlich, da die Server-Komponente von *ping* in nahezu allen netzwerkfähigen Betriebssystemen bereits im Kernel implementiert ist. Somit ist es mit aktiven Messungen möglich, einen oder mehrere Datenströme mit einer gewünschten Verkehrscharakteristik in das Netz zu injizieren und aus der Beobachtung der Sende- und Empfangszeiten einzelner Datenpakete das Verhalten der Datenströme im Netz zu analysieren. Messgrößen, welche in einem aktiven Messszenario ermittelt werden können, sind in Kapitel 3.2.7 aufgelistet.

Passive Messungen kommen nur für Netzbetreiber in Betracht, weil sie einen Zugang zu den Netzknoten im inneren Netz erfordern. Da laut Zielsetzung dieser Arbeit das Messsystem auch den Dienstanbietern und den Nutzern des Netzes Messungen ermöglichen soll, kommen nur aktive Messungen für das zu entwickelnde System in Betracht. Außerdem ist mit aktiven Messungen gegenüber passiven ein wesentlich breiteres Spektrum an möglichen Untersuchungen realisierbar. Es ist aber zu beachten, dass durch aktive Messungen im Netz sonstige Datenströme im Netz gestört werden können.

3.2.3.1 Messen mit unterschiedlichen Quellenmodellen

Bei der Durchführung aktiver Messungen stellt sich unter anderem die Frage des Quellenmodells für das Aussenden der Datenpakete. Häufig werden bei aktiven Messungen Probepakete konstanter Länge in gleichen zeitlichen Abständen gesendet. Ein derartiges Quellenmodell wird als *Constant Bitrate Source (CBR)* bezeichnet. Das Messen mit einer CBR-Source hat folgende Nachteile:

- Die Paketverzögerung sowie die Wahrscheinlichkeit für einen Paketverlust in einem Netzknoten hängt von der Paketlänge ab [Sie00]. Diese Abhängigkeit kann mit der Messung mit Paketen derselben Größe nicht aufgedeckt werden.
- Die Messergebnisse hängen sowohl von der augenblicklichen als auch von der mittleren Datenrate des Mess-Datenstroms ab. Die Behandlung von Datenströmen von burstartigen Quellen kann mit einer CBR-Quelle nicht untersucht werden. Wird mit einer CBR-Quelle z.B. eine geringe Paketrate gewählt, so werden mit einer derartigen Messung die durch den burstartigen Datenverkehr auftretenden kurzzeitigen Überlasten im Netz nicht erfasst. Wird eine hohe Paketrate gewählt, so wird das Netz durch die Messung stark belastet. Damit werden die Ergebnisse ebenfalls verfälscht.
- In verteilten Messszenarien, in welchen mehrere Mess-Datenströme simultan übertragen werden, kann der Fall auftreten, dass mehrere Datenquellen zu gleichen Zeitpunkten die Datenpakete aussenden. Damit tritt eine Synchronisation von Datenströmen im Netz auf. Dieses Problem wird in der Literatur als *Global Synchronization Problem* [Flo93] bezeichnet und kann speziell TCP-Datenströme stark beeinträchtigen. Um das Problem der Synchronisation von Datenströmen zu vermeiden, empfiehlt z.B. die IPPM-Working Group eine Realisierung von Datenquellen mit einer zufälligen Verteilung der Zwischenankunftszeiten der Pakete.

Neben den oben genannten Parametern der Datenquelle hängt die gemessene Dienstgüte in DiffServ-fähigen Netzen (siehe Kapitel 2.2) vom verwendeten DSCP ab.

Für die Übertragung der Anwendungsdaten in IP-Netzen werden fast ausschließlich die Transportprotokolle TCP und UDP verwendet. Dabei reagieren TCP- und UDP-basierte Anwendungen unterschiedlich auf Paketverzögerungen sowie Paketverluste. Damit ist die Dienstgüte einer Anwendung auch vom verwendeten Transportprotokoll abhängig. Um diese Abhängigkeit zu berücksichtigen, sind Messungen mit beiden Transportprotokollen zu realisieren.

Es existieren weitere Transportprotokolle, wie z.B. das *Xpress Transfer Protocol (XTP)* [Str92]. Diese finden aber sehr selten Anwendung und werden nur von weni-

gen Betriebssystemen unterstützt. Aus diesem Grund werden in dieser Arbeit nur die Transportprotokolle TCP und UDP berücksichtigt.

Aus den vorgestellten Nachteilen der CBR-Quelle sowie aus der genannten Abhängigkeit der Messergebnisse von den Charakteristiken des Mess-Datenstromes wird die Notwendigkeit der Betrachtung folgender Parameter festgestellt:

- Verteilung der Zwischenankunftszeiten der Datenpakete
- Verteilung der Paketlängen
- Verwendung eines vorgegebenen DSCP
- Verwendung von TCP und UDP

3.2.3.2 Passive Messungen in dieser Arbeit

Passive Messungen werden benötigt, um bereits existierende Datenströme zu analysieren. Mit den damit gewonnenen Erkenntnissen lässt sich die Parametrisierung und Dimensionierung des Netzes durchführen.

Die in Tabelle 3.1 dargestellte Verteilung der Datenmengen, die im Internet mit unterschiedlichen Transportprotokollen übertragen werden, verdeutlicht die Notwendigkeit der Messung mit TCP-Datenströmen. Die Werte in der Tabelle stammen aus einer Be-

Messgröße	**TCP**	**UDP**	**sonstige**
Anteil am Datenvolumen [%]	83	16	1
Anteil am Paketvolumen [%]	75	22	3

Tabelle 3.1: Verteilung des Verkehrsvolumens nach Transport-Protokollen im IP-Netz des US-amerikanischen Unternehmens Sprint

obachtung der Datenströme im US-amerikanischen IP-Netz des Unternehmens Sprint im Zeitraum zwischen 1998 und 2003 [Fom04]. Diese Ergebnisse decken sich in etwa mit den am G-WiN-Anschluss der Universität Hannover gesammelten Statistiken.

Des Weiteren werden passive Messungen benötigt, um statistische Eigenschaften von Datenströmen einzelner Anwendungen, insbesondere hinsichtlich der Zwischenankunftszeiten der Datenpakete sowie der Paketlängen, zu bestimmen. Neben den bereits existierenden Untersuchungen der Charakteristik einzelner Anwendungen ist es häufig notwendig, Datenpakete einzelnen Datenströmen zuzuordnen, ohne den Aufbau und Abbau der logischen Verbindungen zu verfolgen, und anschließend die Häufigkeitsverteilung der Zwischenankunftszeiten der Datenpakete sowie der Paketlängen zu analysieren.

Dies ist besonders bei Anwendungen relevant, in welchen mehrere logische Verbindungen und entsprechend mehrere Datenströme unterhalten werden. Zu solchen gehören vor allem IP-Telefonie- und Videokonferenz-Anwendungen nach H.323, SIP, aber auch Streaming-Anwendungen und Online-Spiele. Mit den so gewonnenen Häufigkeitsverteilungen können Datenströme einzelner Anwendungen modelliert werden, welche im Netz eine gleiche Behandlung erfahren wie die betrachteten Anwendungen. Für eine beschriebene Extraktion von Datenströmen wurde als begleitende Maßnahme für diese Arbeit ein Werkzeug *FlowStats* entwickelt. Erste Ergebnisse der Untersuchung einer VoIP-Anwendung sind bereits in [Sie03] veröffentlicht.

Eine Beschreibung der Funktion von *FlowStats* ist in Anhang A gegeben. Die Modellierung der Datenströme von Anwendungen kann unabhängig vom Thema Messungen betrachtet werden. Die Untersuchung der Quellenmodelle sowie die Modellierung dieser Quellen nimmt keinen direkten Einfluss auf das Design des Messsystems.

3.2.4 Umlauf-Messungen

Ein häufig für aktive Messungen verwendetes Messverfahren ist die Umlauf-Messung. Der Ablauf einer Umlauf-Messung ist schematisch in Abbildung 3.4 dargestellt. Die

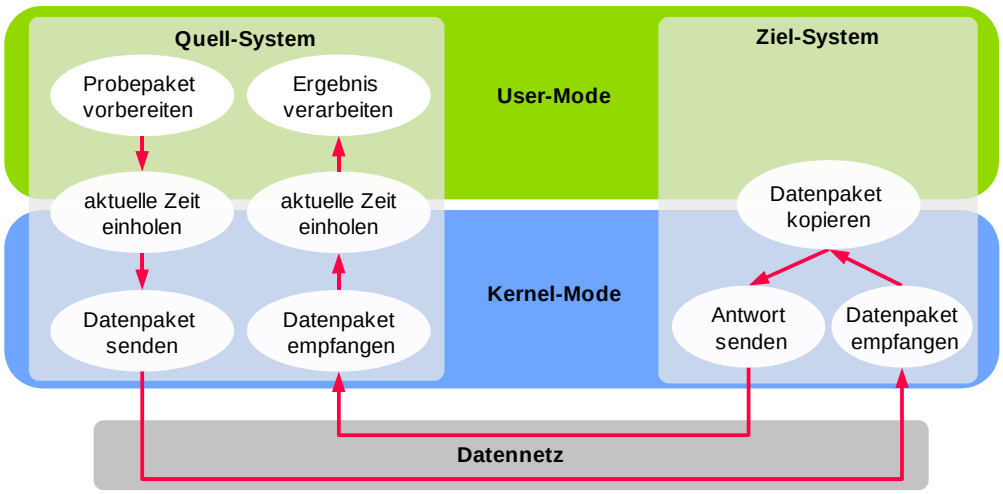

Abbildung 3.4: Ablauf einer Umlauf-Messung

Zuordnung der funktionalen Blöcke zum Kernel- bzw. User-Mode ist dabei so gewählt worden, dass für die Realisierung eines Messsystems keine Eingriffe in den Kernel notwendig sind.

Ein Datenpaket wird beim Sender vorbereitet, wobei die Systemzeit unmittelbar vor dem Aussenden abgefragt wird. Damit wird die Wahrscheinlichkeit für das Unterbrechen des Prozesses zwischen dem Einholen der Zeit und dem Aussenden des Datenpakets minimiert. Zur Vorbereitung des Datenpakets gehört unter anderem die Berechnung der Aussendezeit entsprechend dem gewählten Quellenmodell sowie das Füllen des Datenpakets mit gewünschter Information. Für das Versenden wird das Datenpaket über die Socket-Schnittstelle [Ste97b] an den TCP/IP-Stack des Kernels weitergegeben. Die Verarbeitung auf den Ebenen 1 bis 4 des OSI-Stacks wird dabei vom Kernel des Betriebssystems und von der Hardware übernommen. Beim Empfänger wird auf der Anwendungsebene das Datenpaket kopiert und sofort an die Absenderadresse zurückgesendet.

Der Empfänger-Prozess bzw. Thread liest in der Regel die Daten blockierend vom Socket. Wird ein Datenpaket gelesen, so ist möglichst zeitnah die Systemzeit zu ermitteln. Die Umlaufzeit wird dann als Differenz zwischen der Aussendezeit und der Empfangszeit berechnet.

3.2.4.1 Fehlerabschätzung

Der Fehler bei einer Umlauf-Messung entsteht in erster Linie durch die Verzögerung zwischen dem Bestimmen der Systemzeit und dem tatsächlichen Senden respektive Empfangen des Datenpakets (siehe die Diskussion der Größe τ_{sched} in Kapitel 2.5.2). Eine weitere Fehlerursache liegt in der durch den Schlupf der Uhr verursachten Veränderung des Offsets in der Zeit zwischen dem Senden und dem Empfangen des Datenpakets.

Die Verzögerung der Datenpakete im Netz liegt zwischen wenigen Millisekunden und maximal 2 – 3 s. Die Stabilität der hochaufgelösten Systemuhr hängt von der Frequenz-Stabilität des Prozessor-Taktgebers ab. Dieser hat eine relativ hohe immanente Abweichung der Soll-Frequenz. Nach [Mil96, Kap. 3] sowie [Pas01, Kap.3.1] liegt diese in der Größenordnung von 10^{-5}. Die Variation dieser Frequenz ist aber wesentlich geringer und liegt bei ca. 10^{-7}. Da bei der Umlauf-Messung nur die Differenz der Offsets der Uhr zum Zeitpunkt des Aussendens sowie des Empfangs eines Datenpakets relevant ist, beträgt die Veränderung des Offsets in der Zeitspanne des Empfangs von benachbarten Datenpaketen mit der oben genannten Stabilität der Uhr maximal $20 - 30$ µs. Wird der systematische Fehler, verursacht durch den immanenten Offset der Frequenz – den Schlupf – empirisch ermittelt und eliminiert, so ist nur die Variation der Frequenz für den Messfehler verantwortlich. Mit den oben genannten maximalen Laufzeiten der Datenpakete und der Variation der Frequenz des Taktgebers ergibt sich ein zufälliger Fehler durch die Frequenzschwankung von weniger als 1 µs. Die Fehlerkorrektur kann allgemein wie folgt gemacht werden:

Wird empirisch ein Schlupf s ermittelt und wird eine Messung über einen Intervall T

beobachtet, so sind die gemessenen Werte um den Korrekturwert k, $k = s \cdot T + O$ zu berichtigen. Die konstante Komponente O des Offsets kann z.B. durch einen einmaligen Abgleich auf die Referenz-Uhr zum definierten Zeitpunkt bestimmt werden.

Die Zeit zwischen dem Einholen des Zeitstempels und dem Aussenden eines Datenpakets τ_{sched} (siehe Abbildung 2.5) hängt von der verwendeten Hardware, dem Betriebssystem und der Implementierung des Messsystems ab. Es kann an dieser Stelle keine genaue Abschätzung von τ_{sched} gegeben werden. Vielmehr wird in Kapitel 5.1.4 eine Abschätzung des Fehlers τ_{sched} für das implementierte System empirisch ermittelt. Es wird aber erwartet, dass das Maximum der Fehler τ_{sched} im Bereich ≤ 100 µs liegen wird. Dieser Fehler tritt aber bei Umlauf-Messungen und bei Einweg-Messungen in etwa in gleicher Weise auf. Für die Wahl zwischen den beiden Messszenarien ist die Betrachtung von τ_{sched} also nicht erforderlich.

Bei der Auflösung der Systemuhren wird von einer Auflösung von 1 µs oder höher ausgegangen (siehe Kapitel 3.3) – diese Auflösung bieten alle modernen Unix-Systeme [Get95]. Damit ist die Grenze des Auflösungsfehlers $\tau_{res} = 0,5$ µs. Dieser fällt gegenüber dem erwarteten Fehler τ_{sched} nicht ins Gewicht und kann somit vernachlässigt werden. Damit wird der Fehler bei der Umlauf-Messung im Wesentlichen durch den Fehler τ_{sched} bestimmt.

In der Regel sind die Ergebnisse einer Umlauf-Messung bei der Betrachtung der Dienstgüte nicht direkt von Interesse. Diese dienen lediglich der Abschätzung der entsprechenden Einweg-Größen – der Einweg-Verzögerung und der daraus abgeleiteten Größen inklusive der Einweg-Paketverluste (siehe Kapitel 2.4.1). Dabei wird die Einweg-Verzögerung durch die Halbierung der Umlauf-Verzögerung errechnet. Die Bestimmung der Einweg-Verzögerung durch die Halbierung der Umlauf-Verzögerung ist aber stark fehlerbehaftet.

Eine Ursache dafür ist die Asymmetrie der Verzögerungen. Die wesentlichen Ursachen für die unterschiedlichen Verzögerungen je Richtung sind die Warteschlangen-Verzögerung t_{queue}, die Paketwiederholungsverzögerungen t_{retr} sowie die Ausbreitungsverzögerung t_{prop} (siehe Abbildungen 3.5 und 3.6). Häufig treten größere Warteschlangen-Verzögerungen nur in einer einzigen Queue auf – nämlich an dem am meisten belasteten Punkt im Netz. Damit ergibt sich der Grenzwert für den Fehler bei der Berechnung $\tau_{queue} = \frac{t_{queue}}{2}$. Entsprechendes gilt für t_{retr}. Ein Fehler bei der Berechnung der Ausbreitungsverzögerung tritt auf, wenn im Netz asymmetrisches Routing verwendet wird. Mit den Werten aus Tabelle 2.5 ergibt sich für jeden Kilometer Entfernungsdifferenz der Hin- und Rück-Richtung ein Fehler $\tau_{prop} \approx 1,5 - 3$ µs.

Wird angenommen, dass auf terrestrischen Strecken Entfernungsdifferenzen je Richtung nicht mehr als 1.000 km betragen, so bleibt der Fehler $\tau_{prop} \leq 3$ ms. Kommen in einer Richtung dagegen Satellitenverbindungen zum Einsatz (ca. 36.000 km je Richtung), so

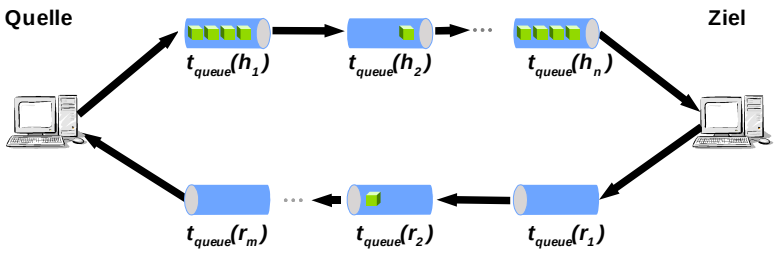

Abbildung 3.5: Messfehler, verursacht durch eine asymmetrische Belastung der Warteschlangen

steigt der Fehler auf $\tau_{prop} \approx 120$ ms. Sind die Komponenten t_{queue} und t_{retr} viel größer als sonstige Verzögerungskomponenten im Netz, so kann der Fehler bei der Abschätzung der Einweg-Verzögerung aus der Umlauf-Verzögerung bis zu 50 % des gemessenen Wertes betragen. Somit ist das Umlauf-Messszenario für die hochpräzise Messung der Einweg-Verzögerungen sowie der daraus abgeleiteten Größen nicht geeignet.

Für eine anschauliche Darstellung der Fehlerabschätzung soll beispielhaft folgende Messung im Umlauf-Messszenario betrachtet werden. Zusätzlich zu den oben dargestellten Annahmen soll die Entfernung zwischen Sender und Empfänger in eine Richtung 1.000 km und in die Gegenrichtung 1.300 km betragen. Die Warteschlangen-Verzögerung soll im Wesentlichen in einer einzigen Queue auftreten und 20 ms betragen. Die Processing-Zeit sei vernachlässigbar gering. Des Weiteren wird angenommen, dass der Fehler τ_{sched} den Wert 100 µs nicht überschreitet und keine Retransmissions auftreten. Damit wird unter Vernachlässigung des Fehlers τ_{sched} folgende Umlauf-Verzögerung D_{RTT} gemessen:

$$D_{RTT} = t_{prop} + t_{queue} = (1.000 + 1.300)\ km \cdot 0,006\ \frac{ms}{km} + 20\ ms = 33,8\ ms$$

Hieraus wird folgende Einweg-Verzögerung D_{OWD} berechnet:

$$D_{OWD} = \frac{D_{RTT}}{2} = \frac{33,8\ ms}{2} = 16,9\ ms$$

Die Fehlerabschätzung für die betrachtete Messung ist in Tabelle 3.2 zusammengefasst. Dabei wird der Verfahrensfehler als τ_{verf} bezeichnet. Der Fehler τ_{offset} kann bei der oben angegebenen Stabilität der Uhren von 10^{-7} und der angenommenen Umlauf-Verzögerung von 33,8 ms vernachlässigt werden. Bei der Messung dominiert der Verfahrensfehler τ_{verf}. Ein Gesamtfehler von 11 ms ist für die Betrachtung der Einweg-Jitter und der Einweg-Verzögerungen zu hoch, so dass ein anderes Verfahren für das Messen zu verwenden ist.

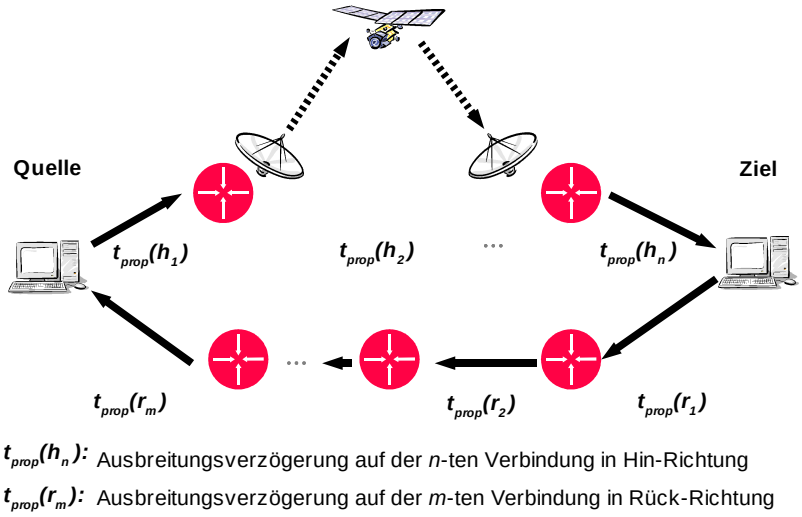

Abbildung 3.6: Messfehler, verursacht durch ein asymmetrisches Routing im Netz

Betrachtete Größe	**Umlauf-Verzögerung**	**Umlauf-Jitter**	**Einweg-Verzögerung**	**Einweg-Jitter**
τ_{sched}, [µs]	100			
τ_{offset}	≈ 0			
τ_{verf}, [µs]	0		$\frac{20.000}{2} + 3 \cdot (1.300 - 1.000) = 10.900$	
τ_{ges}, [µs]	100		11.000	

Tabelle 3.2: Fehlerabschätzung im Umlauf-Messszenario

Ebenso fehlerbehaftet ist die Messung der Paketverluste, da mit diesem Messverfahren nicht feststellbar ist, ob ein Datenpaket auf dem Hin- oder Rückweg verloren gegangen ist.

3.2.5 Einweg-Messungen

Das wesentliche Problem der Umlauf-Messung ist die Abschätzung der Einweg-Verzögerungen aus den gemessenen Umlauf-Verzögerungen (siehe Kapitel 3.2.4). Aus diesem Grund sind – falls möglich – Einweg-Messungen den Umlauf-Messungen vorzuziehen. Das Strukturbild einer Einweg-Messung ist in Abbildung 3.7 dargestellt. Die Abschätzung der Fehler bei Einweg-Messungen wird in Kapitel 3.2.5.2 gemacht.

Der wesentliche Unterschied im Ablauf gegenüber der Darstellung in Abbildung 3.4 liegt darin, dass die funktionalen Blöcke für den Empfang der Datenpakete vom Sender zum

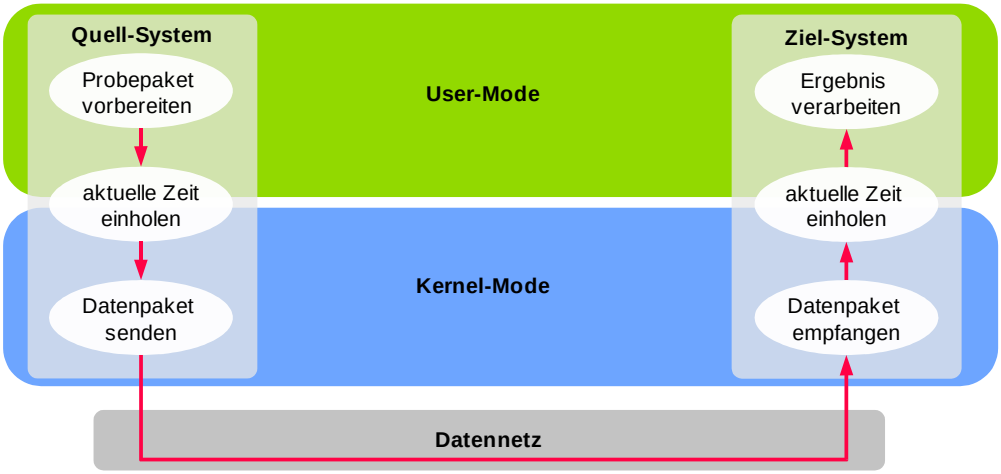

Abbildung 3.7: Ablauf einer Einweg-Messung

Empfänger verlagert werden. Dabei werden dieselben Schnittstellen zum Betriebssystem verwendet wie bei Umlauf-Messungen.

Ein mit Einweg-Messungen verbundenes Problem ist die Synchronisation der Uhren zwischen Sender und Empfänger. Dieses Problem sowie mögliche Lösungen dafür werden nachfolgend vorgestellt.

3.2.5.1 Uhrensynchronisation

Das Problem der Uhrensynchronisation ist bereits ausführlich in anderen Arbeiten behandelt worden [Mil96, Pas01, Smo03]. An dieser Stelle werden lediglich die für die Fehlerabschätzung notwendigen Informationen zusammengefasst.

Bei der Uhrensynchronisation spielen drei Faktoren eine wesentliche Rolle:

1. Die Stabilität des Referenz-Zeitgebers.
2. Die Verfügbarkeit und die Güte der Strecke bzw. des Netzes, über welches die Referenz-Zeit übermittelt wird.
3. Fehler bei der Synchronisierung der lokalen Rechneruhr auf das Referenz-Signal.

Eine hohe Stabilität wird gegenwärtig mit Atomuhren erreicht. Diese haben eine relative Frequenzschwankung von 10^{-13} bis 10^{-14} [Bau03]. Soll eine hohe Güte der Synchronisierung erreicht werden, so ist das Messsystem auf eine Atomuhr zu synchronisieren.

Als Transportnetz für die Übermittlung der Referenz-Zeit kann unter anderem das IP-Netz verwendet werden, über welches auch sonstige Daten übermittelt werden. Dem Uhrenabgleich über das Internet dient das *Network Time Protocol (NTP)* [Mil92]. Neben dem Verfahren zur Übermittlung der Referenz-Zeit beinhaltet das Protokoll Algorithmen zur Eliminierung der Fehler infolge von sporadisch auftretenden hohen Jittern im Netz. Unter Einsatz von NTP kann eine Synchronisation der Rechneruhren auf 1 – 2 ms erreicht werden. Voraussetzungen für eine solche Synchronisation ist eine gute Verbindung zwischen den betrachteten Rechnern und den Time-Servern sowie geringe Jitter [Pas03]. Treten hohe Jitter über längere Zeiträume auf, so sinkt die Güte der Synchronisation. Dies lässt sich aber a posteriori feststellen. Werden zu hohe Jitter bei den NTP-Paketen festgestellt, kann z.B. die Messung wiederholt werden. In [Smo03] wird gezeigt, wie man unter bestimmten Bedingungen eine Synchronisation der Uhren auf weniger als 500 µs erreichen kann. Zu den Randbedingungen gehören dabei beispielsweise die Verwendung eines einzigen Time-Servers, das Einhalten einer geringen Polling-Rate sowie das Vermeiden asymmetrischer Routen.

Mit dem *Global Positioning System (GPS)* [PTB04] wurde vom US-amerikanischen Verteidigungsministerium ein Netz geschaffen, über welches eine Zeit-Synchronisation auf 2 – 3 µs realisierbar ist. Zu den Vorteilen des GPS gehört neben den sehr geringen Fehlern die weltweite Verfügbarkeit des GPS-Netzes. Der wesentliche Nachteil entsteht durch die Verwendung einer hohen Trägerfrequenz von 1,5 GHz. Damit entsteht ein Problem des Empfangs des GPS-Signals innerhalb von Gebäuden. In der Regel benötigen GPS-Empfänger eine Außenantenne.

Neben GPS existieren weitere Systeme zur Übertragung der Referenz-Zeit. Diese sind aber nicht weltweit verfügbar. So ist in Deutschland das DCF77-System weit verbreitet, mit dem eine Uhrensynchronisation auf ca. 50 µs erreicht werden kann [Mei04, Grö04].

Die dritte variable Komponente bei der Uhrensynchronisation ist die Verzögerung zwischen dem Empfang und der Verarbeitung des Referenz-Signals im Kernel des Betriebssystems. Es ergibt sich eine variable Verzögerung durch die Interrupt-Zustellung vom Netzwerk-Interface zum Kernel. Bei der Synchronisation über NTP entspricht diese Verzögerung der Verarbeitungszeit des Datenpakets im TCP/IP-Stack. In beiden Fällen kann eine Synchronisation auf 2 – 3 µs erreicht werden [Ubi01, Pas03]. Wird die Synchronisierung in der Anwendung realisiert, so kann durch das Prozess-Scheduling eine weitere variable Verzögerung von mehr als 50 µs dazukommen.

Zusammenfassend kann festgehalten werden, dass mit aktuellen PC-Systemen über das IP-Netz im Normalfall eine Zeit-Synchronisation auf ca. 1 – 2 ms erreichbar ist. Unter

günstigen Bedingungen kann hier eine Synchronisation auf weniger als 500 µs erreicht werden. Unter Verwendung des GPS-Systems kann eine Synchronisation auf bis zu 2 – 3 µs erreicht werden. Mit DCF77 lässt sich die Uhr auf ca. 50 µs auf die Referenz-Uhr synchronisieren.

3.2.5.2 Fehlerabschätzung

Auf die Abschätzung des Fehlers τ_{sched} treffen die Überlegungen in Kapitel 3.2.4.1 zu. Es ist zu beachten, dass bei der Betrachtung der Berechnung der Einweg-Verzögerung aus der gemessenen Umlauf-Verzögerung in Kapitel 3.2.4.1 lediglich ein relativer Fehler angegeben worden ist. Bei der Fehlerabschätzung der Einweg-Messung ist hingegen ein absoluter Fehler angegeben worden, da dieser nicht vom gemessenen Wert abhängt. Schon bei gemessenen Latenzzeiten im LAN von 2 – 3 ms liegt der maximale Fehler bei der Abschätzung der Einweg-Verzögerung aus einer Umlauf-Messung im Millisekunden-Bereich. Folglich ist eine Einweg-Messung einer Umlauf-Messung vorzuziehen, wenn mehr als eine grobe Abschätzung der Werte notwendig ist. Die geschätzten Fehler im Einweg-Messszenario sind in Tabelle 3.3 zusammengefasst. Für den Fehler τ_{sched} wurde

Betrachtete Größe	**Einweg-Verzögerung**			**Einweg-Jitter**		
τ_{sched}, [µs]	≈ 100					
	GPS	DCF77	NTP	GPS	DCF77	NTP
τ_{offset}, [µs]	3	50	2.000	≈ 0		
τ_{ges}, [µs]	≈ 103	≈ 150	≈ 2.100	≈ 100		

Tabelle 3.3: Fehlerabschätzung im Einweg-Messszenario

derselbe Wert angenommen wie bei der Betrachtung der Umlauf-Verzögerung. Die Werte für τ_{offset} ergeben sich aus der Güte der Uhrensynchronisation einzelner Verfahren (siehe Kapitel 3.2.5.1). Bei der Betrachtung der Jitter kann der Fehler τ_{offset} ebenso wie bei den Umlauf-Messungen vernachlässigt werden, da bei einer Stabilität der Uhren von 10^{-7} und einer maximalen Zwischenankunftszeit der Datenpakete von unter einer Sekunde die Differenz der Offsets in der Größenordnung von maximal 100 ns liegt.

3.2.5.3 Weitere Aspekte der Einweg-Messung

Bei der Einweg-Messung sind nicht die Offsets der Uhren an den beteiligten Messsystemen, sondern vielmehr die Differenz der Offsets der Uhren von Bedeutung. Können die Offsets als konstant über die Zeit angenommen werden, so lässt sich die Differenz

mit geringem Aufwand durch zwei Messungen in der Gegenrichtung eliminieren. Das Eliminieren des konstanten Anteils des Offsets ist in [Pax98b, Kap. 4.1] ausführlich beschrieben. Damit lässt sich ohne einen GPS-Empfänger der relative Offset auf wenige Mikrosekunden genau bestimmen.

Außerdem ist zu beachten, dass eine Reduktion der Messfehler mit Hilfe von NTP- bzw. DCF77-Synchronisation nur für Langzeit-Messungen – also Messungen im Bereich von Minuten oder länger – erreichbar ist. Bei Kurzzeit-Messungen kann die Wirkung des NTP-Daemons sich sogar negativ auf die Messfehler auswirken, da der Abgleich auf den externen Zeitgeber eine ständige Justierung der Frequenz der Systemuhr mit sich bringt.

Bei der Berechnung der Jitter werden die Laufzeiten benachbarter Datenpakete betrachtet. Hierfür ist die Stabilität der Uhren-Oszillatoren ausreichend genau, um die Drift der Uhren in einem Zeitintervall zwischen dem Aussenden von zwei Datenpaketen sehr gering zu halten. Beträgt der Abstand zwischen zwei Datenpaketen weniger als 1 s, so liegt der Fehler, verursacht durch den Schlupf und die Drift der Uhren, mit der in Kapitel 3.2.4.1 angegebenen Stabilität der Oszillatoren von 10^{-7}, unter 100 ns und kann somit gegenüber τ_{sched} vernachlässigt werden.

3.2.6 Verteiltes Messen

Aus den in Kapitel 3.1 vorgestellten Messszenarien ergibt sich der Bedarf einer Realisierung verteilter Messungen in IP-Netzen. Es ist eine Tendenz erkennbar, dass die Messung der Netzwerk-Performance von den ISPs oder ASPs zunehmend als ein verteilter Dienst im IP-Netz angeboten wird [Geo01, Hol04]. Dabei können Kunden bestimmte Parameter der Dienstgüte zwischen einem Punkt im Internet zu dem Aufpunkt des Anbieters messen.

3.2.6.1 n:1-Messszenario

Ein Szenario, bei welchem mehrere Teilnehmer die Dienstgüte zu einem bestimmten Aufpunkt im Netz messen können, wird in dieser Arbeit ein *n:1*-Szenario (siehe Abbildung 3.8) genannt. Ein *n:1*-Messszenario ist in erster Linie für die Bestimmung der Güte der Erreichbarkeit von bestimmten Servern im Internet geeignet. Dabei können mehrere Nutzer die Dienstgüte gegenüber einem in der Nähe des Servers platzierten Messsystems überprüfen bzw. überwachen. Es existieren folgende Möglichkeiten, ein verteiltes *n:1*-Szenario zu realisieren:

1. Ein Mess-Server ist gestartet. Clients können sequentiell Mess-Sessions durchführen. Ein simultaner Zugriff auf den Server ist nicht möglich. Diese Lösung ist

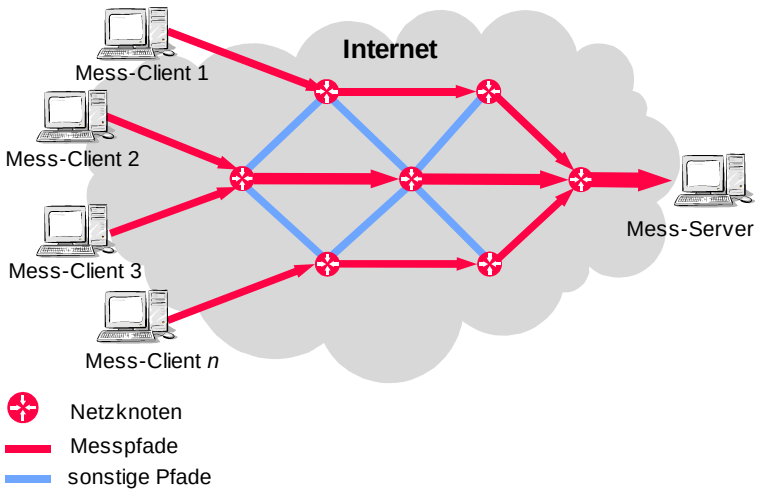

Abbildung 3.8: Verteiltes 1:n-Messszenario

mit bereits existierenden Messtools (siehe Kapitel 2.7) realisierbar. Die Lösung skaliert aber unzureichend. Es werden zusätzliche Mechanismen zur Reservierung des Mess-Servers benötigt.

2. Für jede neue Mess-Session wird eine neue Instanz des Mess-Servers gestartet. Diese Lösung ist ebenfalls mit bereits existierenden Tools realisierbar. Es fehlen aber Mechanismen, um für jede Client-Anfrage einen neuen Server-Prozess zu starten. Bei Messungen mit hohen Datenraten stören sich einzelne Datenströme gegenseitig. Es sind keinerlei Mechanismen zur Vermeidung solcher Störungen vorgesehen.

3. Der Mess-Server ist multisession-fähig. Es können mehrere Clients simultan bedient werden. Es findet keine Synchronisation zwischen einzelnen Mess-Sessions statt. Damit ist das Problem des Starts mehrerer Server-Instanzen für simultane Messungen behoben. Derartige Systeme werden mit Hilfe von Multithread-Techniken [Ste97b] realisiert. Hochperformance-Messungen sind aber nur bedingt möglich, da einzelne Server-Threads sich gegenseitig stören und damit die Messergebnisse verfälschen.

   Eine derartige Multisession-Fähigkeit ist z.B. seit kurzem in Iperf verfügbar. Diese ist aber speziell für die Untersuchung der gegenseitigen Verdrängung konkurrierender TCP-Ströme implementiert worden [Tir04]. Iperf ist in erster Linie für die Messung des Durchsatzes entwickelt worden. Eine hochpräzise Messung der Paketverzögerungen sowie der Jitter ist damit nicht möglich.

4. Der Mess-Server ist multisession-fähig. Die Generierung neuer Session-Threads

wird dabei von einer Hauptroutine kontrolliert. Soll von einem Client eine besonders präzise Messung oder eine Messung mit hoher Datenrate durchgeführt werden, so wird eine blockierende Session angefordert. In diesem Fall wird vom Server sichergestellt, dass eine einzige Messung zum selben Zeitpunkt auf dem Server zugelassen wird. Ist dies nicht möglich, da andere Sessions auf dem Server bereits laufen, wird die blockierende Anfrage abgewiesen.

Mit diesem Ansatz kann eine skalierbare verteilte *n:1*-Messumgebung realisiert werden, ohne auf die Möglichkeit zu verzichten, bei Bedarf eine einzelne hochpräzise Messung auf dem Mess-Server durchzuführen.

Der letzte Ansatz bietet eine größere Flexibilität sowie eine bessere Skalierung als die anderen Möglichkeiten. Aus diesem Grund wird er in dieser Arbeit weiter verfolgt.

3.2.6.2 1:m-Messszenario

Ein Messszenario, in dem von einem Client im IP-Netz zu mehreren Mess-Servern gemessen wird (siehe auch Abbildung 3.3), wird in dieser Arbeit als ein *1:m*-Messszenario bezeichnet.

Mit zunehmender Vielfalt der Dienste, die in IP-Netzen realisiert werden, gewinnt dieses Szenario an Bedeutung. Mit diesem Messszenario kann ein Nutzer von Diensten im Internet die Güte der Erreichbarkeit unterschiedlicher Server überprüfen bzw. überwachen. Mit den gewonnenen Ergebnissen kann anschließend eine Entscheidung getroffen werden, von welchem Server der gewünschte Dienst in Anspruch genommen wird. Bei einigen Anwendungen, wie z.B. dem Grid-Computing, besteht sogar der Bedarf, kontinuierlich die Parameter des Netzes von einem Rechner-Knoten zu mehreren zu überwachen, um in der Lage zu sein, dynamisch den Datenaustausch sowie die Rechenaufgaben über die Knoten zu verteilen.

3.2.6.3 n:m-Messszenario

Ein verteiltes $n : m$-Messszenario ergibt sich aus einer Verallgemeinerung des $n : 1$ und des $1 : m$ Messszenarios. Dabei wird simultan zwischen n Clients und m Servern die Dienstgüte bzw. die Netzwerk-Performance gemessen. Der Bedarf nach einem solchen Messszenario ergibt sich direkt aus der Betrachtung der Netzmessungen für die Netzbetreiber (siehe Abbildung 3.1.

Da die Mess-Clients direkt von den Nutzern gesteuert werden, kann ein *m:n*-Messszenario erreicht werden, indem mehrere Clients simultan n:1-Messungen starten.

Dabei werden Mechanismen benötigt, um bei Bedarf eine gewünschte zeitliche Beziehung zwischen einzelnen Mess-Sessions herzustellen. Dieser Aspekt wird in Kapitel 3.4.2 genauer beleuchtet.

3.2.7 Messbare Größen

Bei der Diskussion möglicher Messszenarien ist gezeigt worden, dass mit aktiven Einweg-Messungen die Zielsetzung dieser Arbeit am besten erreicht werden kann. In diesem Szenario ist es möglich, folgende Messgrößen zu bestimmen:

- Einweg-Verzögerung
- Jitter
- Einweg-Paketverlust
- Umlauf-Verzögerung
- Umlauf-Paketverlust
- Vertauschung der Reihenfolge der Ankunft von Datenpaketen (*engl. Packet Reordering*)
- Durchsatz (BTC)

Des Weiteren kann mit dem gewählten Szenario unter Einbeziehung gewisser statistischer Methoden die Datenrate auf unterschiedlichen Teilstrecken zwischen Sender und Empfänger sowie die Auslastung der langsamsten Verbindung abgeschätzt werden. Solche Methoden werden in [Nav03] und [Dov01] vorgestellt.

Die Umlauf-Verzögerung kann entweder in einem Umlauf-Messszenario gemessen oder aus zwei entgegengerichteten Messungen der Einweg-Verzögerung berechnet werden (siehe Kapitel 2.3.2). Damit stellt sich die Frage, ob bei der Realisierung des Messsystems eine Möglichkeit zur direkten Messung der Umlauf-Verzögerung sowie der Umlauf-Paketverluste zu implementieren ist. In [Low02] wird beispielsweise empfohlen, die Güte von File-Transfers für Grid-Anwendungen in einem Umlauf-Szenario zu bestimmen. Dieses erscheint aber aus folgenden Gründen nicht zweckmäßig:

- Speziell bei großen Datentransfers über TCP sind die Verteilungen der Paketlängen je Richtung unterschiedlich. In die Vorwärts-Richtung werden viele Pakete voller oder nahezu voller MTU-Größe übertragen. In der Rückwärts-Richtung werden nur Bestätigungen (TCP-ACKs) übertragen. Erfahrungsgemäß haben diese IP-Pakete eine Länge von 40 und 44 Byte. Kurze und lange Datenpakete werden aber im Netz unterschiedlich behandelt.

- Im TCP kann mit einem Antwort-Paket der Empfang von mehreren Datenpaketen bestätigt werden. Damit unterscheidet sich die Paketrate der Datenströme je Richtung.

Somit ist ein Datentransfer über TCP mit einer Einweg-Messung besser als mit einer Umlauf-Messung abgebildet. Der unidirektionale Datenstrom entspricht dabei dem File-Transfer bei Grid-Anwendungen. In der Gegenrichtung wird ein Datenstrom von Bestätigungen übertragen, welcher die Messergebnisse der Einweg-Messung in beiden Fällen in gleicher Weise beeinflusst. Somit ist eine Implementierung eines Umlauf-Messszenarios nicht erforderlich.

3.3 Performance und die zeitliche Auflösung des Messsystems

Im Mittelpunkt dieser Arbeit stehen Ende-zu-Ende-Messungen, deren Charakteristiken durch die Anforderungen betrachteter Anwendungen bestimmt werden. Die sich aus der Zielsetzung ergebenden Anforderungen an die Performance sowie an die notwendige zeitliche Auflösung des Messsystems sind an dieser Stelle zu bestimmen.

Bei der Festlegung der maximalen Datenrate werden folgende Fakten berücksichtigt:

- Moderne LANs werden überwiegend auf der Ethernet-Technologie nach IEEE 802.3 [IEEE02a] implementiert. Dabei werden Link-Datenraten von bis zu 1 Gbit/s verwendet.
- In Weitverkehrsnetzen auf der Basis von SDH [ITU97a] oder 10-Gigabit-Ethernet nach IEEE 802.3ae [IEEE02b] werden Datenraten oberhalb von 1 Gbit/s erreicht. Derartige Strecken werden im Backbone verwendet.
- Es existieren zurzeit nur einzelne Anwendungen, die eine Ende-zu-Ende-Datenrate von mehr als 1 Gbit/s erreichen. Dennoch soll die vom Messsystem theoretisch erreichbare Datenrate oberhalb der Grenze von 1 Gbit/s liegen, um für ein weiteres Wachstum der Performance der Anwendungen gerüstet zu sein. Sie darf unter 10 Gbit/s liegen, da diese Datenraten von modernen Endsystemen noch nicht erreicht werden.

Eine Paketrate von deutlich mehr als 100.000 Pakete/s soll erreichbar sein, um Fast-Ethernet- sowie Gigabit-Ethernet-Strecken mit der maximalen Paketrate füllen zu können (siehe Tabelle 2.4).

Die Anforderung an die Auflösung der Zeitmessung ergibt sich aus folgenden Überlegungen:

- Die Auflösung der Zeit soll ausreichend hoch sein, um die von den Anwendungen wahrgenommene Veränderung der Dienstgüte zu quantifizieren.
- Zeitkritische Anwendungen reagieren sensibler auf hohe Jitter als auf hohe Paketverzögerungen. Für die Festlegung der Auflösung ist somit die Messung von Jittern maßgeblich. Jitter von wenigen Millisekunden beeinträchtigen die Güte der Wiedergabe von Audio- und Video-Strömen. Bei IP-Telefonen werden beispielsweise Jitter-Buffer von 5 bis 30 ms verwendet. Dabei ist man bestrebt, Jitter-Buffer so klein wie möglich zu halten. Um Effekte der Füllung der Jitter-Buffer beurteilen zu können, muss die Auflösung der Zeit viel geringer sein als die Zeit-Kapazität eines kleinsten verwendeten Jitter-Buffers. Somit muss die Auflösung im Mikrosekunden-Bereich liegen.
- In Datennetzen mit einer Übertragungsgeschwindigkeit von bis zu 1 Gbit/s liegen die minimalen zeitlichen Abstände zwischen zwei Datenpaketen im Bereich von 3 µs (siehe Tabelle 2.4). Möchte man die Burstiness der Datenpakete untersuchen, so muss die Auflösung im unterem Mikrosekunden-Bereich liegen.
- Auf vielen Unix-Systemen ist der Zugriff auf die Systemzeit mit einer Auflösung höher als eine Mikrosekunde möglich. Der Aufruf ist aber nicht portabel. Außerdem liegt die minimale Unsicherheit der Zeitmessung mit den verwendeten Systemen im Bereich von wenigen Mikrosekunden. Folglich würde eine Auflösung im Nanosekunden-Bereich derzeit keinen erkennbaren Gewinn hinsichtlich der Güte der Messungen bringen, dafür aber einen erhöhten Speicherbedarf sowie eine höhere Prozessor-Last verursachen.
- Der portable Systemaufruf *gettimeofday()* hat eine Auflösung von 1 µs. Wird beim Messtool eine Auflösung von 1 µs verwendet, so kann das Ergebnis dieses Systemaufrufs unverändert für die Auswertung verwendet werden.

3.4 Steuerung des Messsystems

3.4.1 Kommunikation zwischen einzelnen Instanzen

Aktive Messungen können in einer Client-Server-Architektur realisiert werden. Dabei steuert der Client den Messvorgang. Im einfachsten Fall spielt der Mess-Client die Rolle des Senders, welcher Probepakete zum Datenempfänger sendet, der hinsichtlich der Verbindungssteuerung als Mess-Server agiert. Neben der Verbindung zum Austausch von Probepaketen wird eine Kontrollverbindung zwischen dem Client und dem Server

benötigt. Über diese wird unter anderem vor Beginn einer Messung auf der Empfängerseite ein Listener gestartet und dem Client die IP- und Transport-Adresse des Listeners mitgeteilt.

Die Kontrollverbindung wird über TCP realisiert. Hierbei kann der Einsatz geeigneter Middleware Vorteile einer einfachen Erweiterung des Kommunikationsprotokolls zwischen dem Client und dem Server mit sich bringen.

Die Steuerung des Mess-Servers erfolgt bis auf den Start und das Schließen des Server-Programms über den Client. Oft ist es aber wünschenswert, die Performance nicht in Richtung vom Client zum Server, sondern in die Gegenrichtung zu messen. Zu diesem Zweck muss die Möglichkeit geschaffen werden, vom Mess-Server zum Rechner, auf dem der Mess-Client läuft, zu messen. Die Realisierung einer solchen Messung wird in Kapitel 4.2.2 vorgestellt.

3.4.2 Kontrolle beim Programmstart

Das Messsystem soll sowohl über die Angabe von Kommandozeilen-Parametern als auch über eine Konfigurationsdatei gesteuert werden. Über diese Konfiguration soll unter anderem zwischen verfügbaren Quellenmodellen gewählt werden, und es soll damit die Parametrisierung der gewählten Datenquelle erfolgen.

Für die Konfiguration bietet sich die Verwendung der *Extensible Markup Language (XML)* [W3Ca] an. XML ist eine vom *World Wide Web Consortium (W3C)* [W3C04] standardisierte Sprache, welche zunehmend für Konfigurationsdateien verwendet wird.

In verteilten Messszenarien kann es erforderlich sein, definierte zeitliche Verhältnisse zwischen einzelnen Datenströmen festzulegen. Dies kann am folgenden Beispiel dargestellt werden:

In Messungen zwischen zwei Sendern zu einem Empfänger soll der gegenseitige Einfluss von zwei UDP-Datenströmen untersucht werden. Dafür wird ein *n:1*-Messaufbau gemäß Abbildung 3.8 aufgebaut. Nach dem Start des Programms verbindet sich der Mess-Client mit dem Server, um die Parameter der Messung auszutauschen. Durch den manuellen Start der Programme werden die beiden Instanzen der Messung mit einem gewissen zeitlichen Versatz gestartet. Beginnt das früher gestartete Programm direkt nach dem Aushandeln der Parameter mit der Übertragung der Probepakete, so verdrängt der UDP-Datenstrom die zweite Kontrollverbindung. Damit wird das Aushandeln der Parameter der zweiten Mess-Verbindung erst nach dem Beenden des ersten Mess-Datenstroms abgeschlossen. Der Unterschied zwischen dem gewünschten und dem reellen zeitlichen Ablauf solcher Messung ist in Abbildung 3.9 dargestellt. Farblich hervorgehoben sind dabei die unterschiedlichen Phasen der Messung.

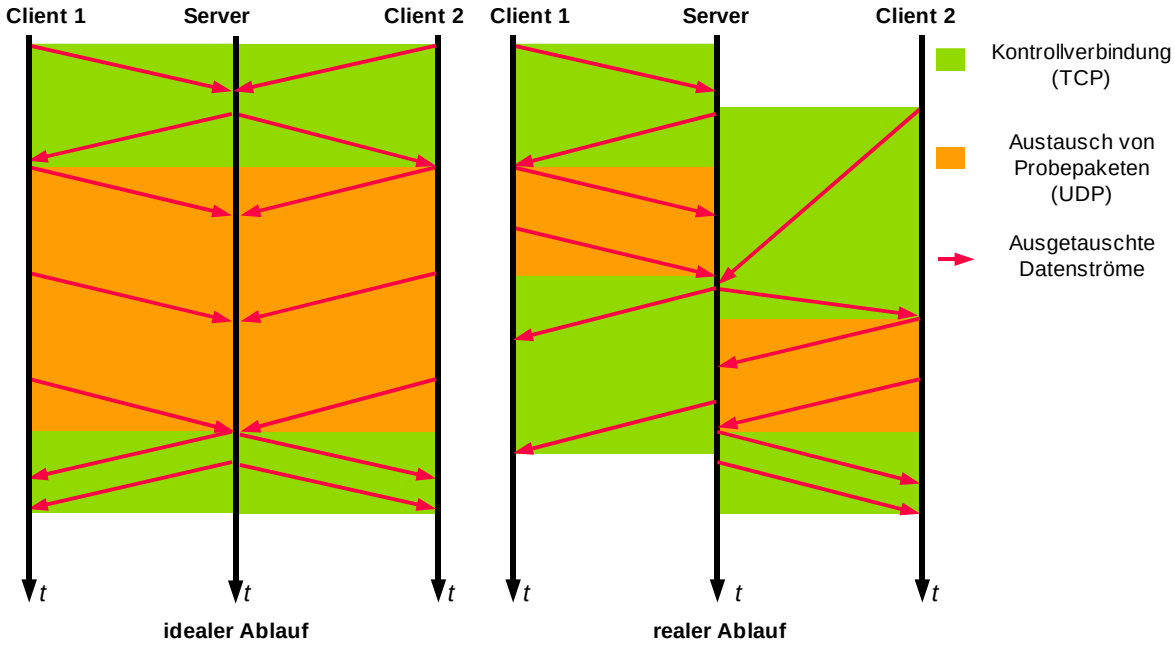

Abbildung 3.9: Zeitlicher Ablauf von Messungen mit konkurrierenden Datenströmen

Um den oben skizzierten Effekt zu vermeiden, soll es möglich sein, den Mess-Datenstrom zeitversetzt zu starten. Damit kann für beide Messungen ein gleicher Zeitpunkt für den Beginn des Austauschs von Probepaketen angegeben werden. Die Parameter für beide Datenströme der Probepakete können dabei im Voraus ausgetauscht werden. Zum vorgegebenen Zeitpunkt können anschließend die konkurrierenden Datenströme gestartet werden. Der zeitliche Ablauf einer solchen Messung ist in Abbildung 3.10 dargestellt. Ebenso kann es in vielen Messszenarien hilfreich sein, den Zeitpunkt für das Beenden der Übertragung von Probepaketen oder die Dauer einer Mess-Session explizit anzugeben.

Wird eine Prozess-Priorisierung für das Messsystem vorgenommen, so muss die Prozess-Priorität ebenso über die Kommandozeile bzw. über die Konfigurationsdatei steuerbar sein, damit die Prozess- bzw. Thread-Priorität fein einstellbar ist. Damit kann unter anderem sichergestellt werden, dass der NTP-Daemon immer noch eine höhere Priorität erhält als das Messsystem, um eine Langzeit-Stabilität der Systemzeit zu erreichen.

3.5 Weiterverarbeitung der Messergebnisse

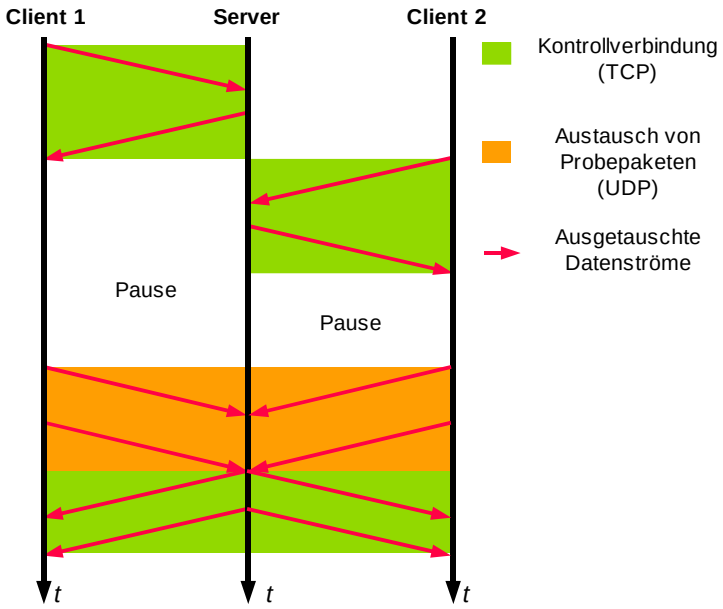

Abbildung 3.10: Auswirkung des zeitversetzten Starts des Datensenders

Neben den direkt in einer Messung ermittelten Sende- sowie Empfangszeiten von Datenpaketen werden weitere abgeleitete Größen berechnet. Die wichtigsten Größen sind in Kapitel 3.2.7 genannt worden.

Häufig werden konkrete Parameter einer Messung in einem iterativen Prozess bestimmt. Dafür ist es erforderlich, einen geringen Satz statistischer Auswertungen der Messergebnisse möglichst direkt nach Abschluss der Messung zur Verfügung zu haben, damit anschließend der nächste Iterationsschritt vorgenommen werden kann. Werden die richtigen Parameter der Messung bestimmt, so können die Ergebnisse einer ausführlichen Analyse unterworfen werden.

Diese Analysen können rechenintensiv sein und sich je nach Messszenario und Problemstellung unterscheiden. Hieraus ergeben sich folgende Anforderungen hinsichtlich der Möglichkeit zur Verarbeitung der Ergebnisse:

- Direkt nach Abschluss einer Messung soll ein minimaler Satz von Statistiken der Messergebnisse vom Programm ausgegeben werden.
- Für eine ausführliche Analyse der Messergebnisse soll nach Abschluss der Messung ein Zugriff auf die Attribute aller gesendeten und empfangenen Datenpakete möglich sein.

- Wegen der Vielseitigkeit der Anforderungen an die Analyse der Ergebnisse soll diese vom eigentlichen Messsystem entkoppelt sein.

Die Anforderungen an die Visualisierung der Messergebnisse ergeben sich aus den Ausführungen in Kapitel 2.6.4. Es ist von Vorteil, wenn aus den gespeicherten Ergebnissen einer Messung direkt ein Satz von Vektorgraphiken – z.B. im SVG-Format[W3Cb, Bro02, W3Cc] – mit den benötigten Verteilungsfunktionen erzeugt wird. Außerdem wird eine tabellarische Ausgabe der Verteilungen benötigt, damit diese anschließend mit externen Programmen wie z.B. *Gnuplot* oder *Microsoft Excel* weiterverarbeitet und visualisiert werden können.

3.6 Realisierung unterschiedlicher Quellenmodelle

Bei dem Entwurf des Messsystems ist es zweckmäßig, ausgewählte Datenquellen direkt in das Programm zu integrieren, und zusätzlich eine einfache Schnittstelle zu schaffen, über welche weitere Quellen zugeschaltet werden können, ohne den Code des Messsystems anpassen zu müssen. Ebenfalls von Vorteil ist es, wenn die Berechnung der Quellenmodelle vom Messsystem entkoppelt werden kann, damit bei Bedarf komplexe Berechnungen der Quellenmodelle auf separate Prozessoren bzw. auf separate Hardware ausgelagert werden kann.

Zum festen Bestandteil des Messsystems soll die am weitesten verbreitete Constant-Bitrate-Quelle (CBR-Quelle) sowie die Datenquelle mit einer Exponentialverteilung der Zwischenankunftszeiten gehören. Die Letztere wird unter anderem im Umfeld der IPPM Working Group verwendet, um das Problem der Synchronisierung mit anderen periodischen Datenströmen zu vermeiden (siehe Kapitel 3.2.3.1).

Kapitel 4

Design und Implementierung von LTest

Für die Realisierung verteilter aktiver Einweg-Messungen wurde im Rahmen dieser Arbeit ein Programm namens *LTest* prototypisch entwickelt. Die darin realisierten Konzepte ergeben sich aus den in Kapitel 3 dargestellten Anforderungen an ein Messsystem. In diesem Kapitel wird das Design sowie die Implementierung des Systems beschrieben. Anschließend werden in Kapitel 5 eine Evaluierung des Systems gegeben und einige typische Messszenarien vorgestellt.

Das Programm wird zurzeit in unterschiedlichen Arbeiten und Projekten eingesetzt (z.B. in [Bie04] sowie in internen Projekten zur Untersuchung der Netzwerk-Performance), und es ist davon auszugehen, dass *LTest* weiter entwickelt wird. Damit ist zu erwarten, dass gerade die Bedienung des Systems noch häufigen Änderungen unterworfen sein wird. Der Quellcode des Programms ist unter der BSD-Lizenz [OSI04] freigegeben. Die Dokumentation und der Quellcode werden unter dem Software-Projekt *LTest* vom Lehrgebiet Rechnernetze der Universität Hannover [Sie04b] gepflegt.

4.1 Grundsätzliche Design-Aspekte

Als übergeordnetes Modell für die Implementierung von *LTest* wurde ein Client-Server-Modell gewählt. Um die Installation und Bedienung des Systems einfach zu halten, wurde das Programm in einer einzigen binären Datei realisiert. Über die Konfiguration kann beim Programmstart angegeben werden, ob die gestartete Instanz als Server oder als Client agieren soll. Alle Parameter können entweder als Kommandozeilen-Parameter oder über eine XML-Konfigurationsdatei angegeben werden.

Um eine einfache Erweiterung des Kommunikationsprotokolls zur Steuerung des Systems zu ermöglichen, kommt eine Middleware zum Einsatz. Bei einer dynamischen Bindung benötigt *LTest* neben der Middleware die Programmbibliothek LibXML2 [XML04] in der Version 2.6 oder höher, welche in allen gängigen Unix-Distributionen zur Verfügung steht. Da diese Programmbibliothek nicht direkt für den Austausch der Probepakete verwendet wird, ist die Güte und die Performance des Systems unabhängig von der Art der Bindung.

4.1.1 Verwendete Hard- und Software

Die Wahl der Hardware und des Betriebssystems für die Erprobung des Systems findet unter den in Kapitel 3.2.1 dargestellten Gesichtspunkten statt. Obwohl das System nur unter Linux mit den Kernel-Versionen 2.4 und 2.6 und Solaris in den Versionen 8 und 9 getestet wurde, ist davon auszugehen, dass es ggf. mit geringen Anpassungen auf sonstigen Unix-Derivaten lauffähig ist, falls die benötigten Bibliotheken auf diesen Systemen installiert sind.

4.1.1.1 Programmiersprache und Programmierumgebung

Ein Ziel dieser Arbeit ist es, hochpräzise Netzmessungen auf Standard-Hardware mit möglichst hoher Performance zu ermöglichen. Aus Performance-Gründen kommen Interpreter-Sprachen ebenso wenig wie die Programmiersprache Java in Betracht. Bei Java wäre eine Ressourcenbindung an das Messsystem während der Garbage-Collection-Phasen nicht gegeben.

Wegen der komplexen Kommunikationsstruktur für die Realisierung verteilter Messungen (siehe Kapitel 4.2) soll eine objektorientierte Sprache verwendet werden. Somit fällt die Wahl auf die Programmiersprache C++. Diese ermöglicht eine Realisierung komplexer verteilter Schnittstellen, unter anderem unter Verwendung von Middleware.

Als Compiler wird sowohl unter Linux als auch unter Solaris der GNU-C++-Compiler [GCC04] in der Version 3.3 verwendet. Dieser gilt als stabil und ausreichend effizient. Kommerzielle Compiler werden hier nicht verwendet.

4.1.1.2 Auswahl geeigneter Middleware

Der wesentliche Aspekt bei der Wahl einer geeigneten Middleware war die Performance sowie die Verfügbarkeit einer freien Implementierung. Dafür kommen prinzipiell SOAP [W3Cd] und CORBA [OMG04] in Betracht. In CORBA werden die über das Netz übertragenen Nachrichten vielfach effizienter codiert als im XML-basierten SOAP. Es existiert eine Reihe freier Implementierungen von CORBA.

Für die Implementierung von *LTest* wurde die quelloffene Implementierung des CORBA-Standards in der Version 2.3 namens MICO [MICO03] verwendet. Diese gilt als eine der stabilsten und am weitesten verbreiteten freien CORBA-Implementierungen.

4.2 Allgemeine Kommunikationsstruktur

Für die Messung mit *LTest* wird genau eine Client- und mindestens eine Server-Instanz benötigt. Die Kommunikation zwischen Client und Server ist bidirektional. Der logische Informationsfluss ist in Abbildung 4.1 dargestellt. Dieser läuft wie folgt ab:

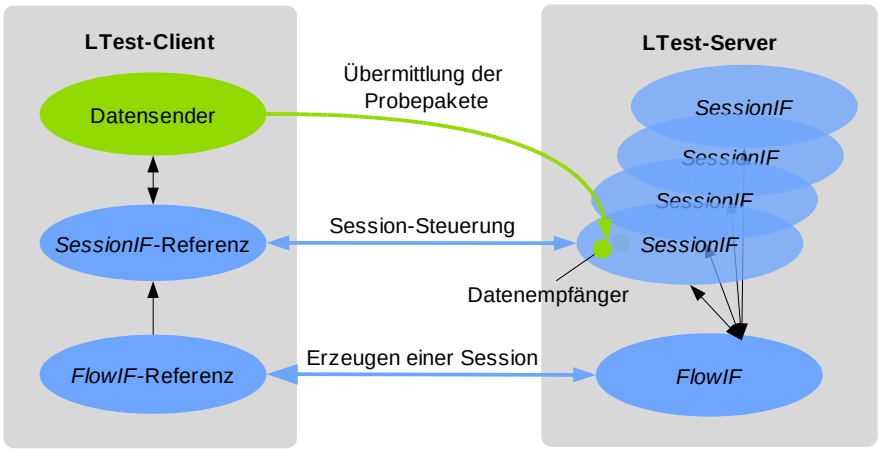

Abbildung 4.1: Grundlegendes Kommunikationsmodell von *LTest*

- Der *LTest*-Client steuert den Server.
- Der Server liefert die Messergebnisse an den Client zurück.
- In Ausnahmefällen werden Nachrichten vom Server zum Client gesendet – z.B. wenn eine Blockierung des Servers nicht möglich ist (siehe Kapitel 4.2.1).
- Im einfachsten Szenario werden die Probepakete vom *LTest*-Client zum *LTest*-Server gesendet. Ein erweitertes Szenario wird in Kapitel 4.2.2 eingeführt.

Das Strukturbild der CORBA-Kommunikation in *LTest* ist in Abbildung 4.2 dargestellt.

Die Steuerung des LTest-Servers erfolgt, indem der CORBA-Client parametrisierte Methoden auf dem CORBA-Server ausführt. Die Ergebnisse der Methodenaufrufe im Server werden ausschließlich als Rückgabewerte an den Client zurückgegeben. Nachrichtenübertragung vom Server zum Client wird über den Exception-Mechanismus von CORBA realisiert. Durch diese Festlegung der Kommunikationsstruktur wird Folgendes erreicht:

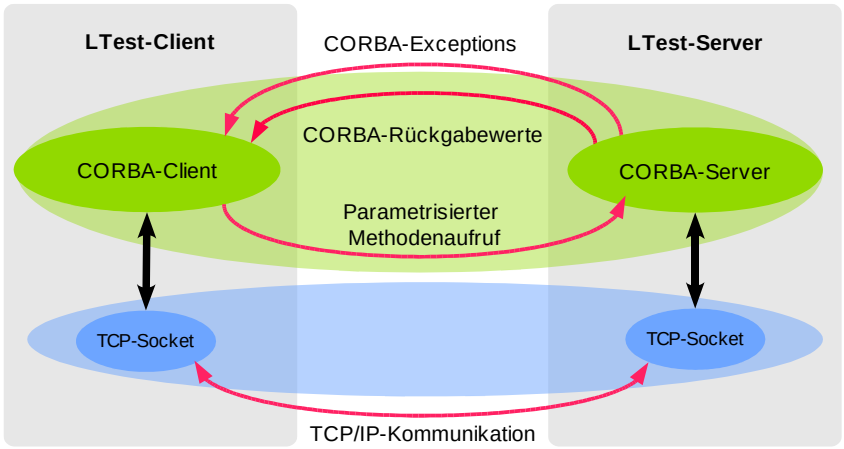

Abbildung 4.2: Strukturbild der CORBA-Communication in *LTest*

- Die Ergebnisse der Methodenaufrufe im Server können nur an den vom Client bestimmten Programmpunkten bei dem Client eintreffen. Es wird kein separater synchronisierter Thread für den Empfang von Daten benötigt, wodurch eine höhere Güte des Datensenders erreichbar ist.
- Die Verwaltung der CORBA-Instanzen sowie der Objekt-Referenzen wird vereinfacht, da jede *LTest*-Instanz nur eine CORBA-Instanz hostet.

Um speziell im *LTest*-Server die Durchführung verteilter Messung zu ermöglichen, wurde dessen Steuerung zweistufig realisiert (vgl. Abbildung 4.1). Nach dem Start des *LTest*-Clients wird in der ersten Stufe nach einem Objekt namens *FlowIf* beim CORBA-Server gesucht. Jeder *LTest*-Server beherbergt genau ein *FlowIf*-Objekt. Dieser dient der Initialisierung der Kommunikation, der Erzeugung neuer Objekte sowie der Synchronisation zwischen einzelnen Objekt-Instanzen. Dabei wird die Adresse des CORBA-Servers über die Kommandozeile oder über die Konfigurationsdatei angegeben. Eine im Quellcode fest codierte Objekt-ID wird gewählt, um eine Abhängigkeit vom CORBA Naming Service zu vermeiden.

Wird unter der in der Konfiguration angegebenen Adresse ein *FlowIf*-Objekt gefunden, so wird die zweite Stufe der Kommunikation eingeleitet. In dieser wird eine Session initiiert, in die die eigentliche Messung eingebettet wird. Die Session ist als CORBA-Objekt namens *SessionIf* realisiert und wird von *FlowIf* erzeugt. Für jede Session wird ein separater Programm-Thread gestartet. In *FlowIf* werden die Referenzen zu den erzeugten Sessions gespeichert, um eine Synchronisation zwischen einzelnen Sessions zu ermöglichen.

Der Rest der Messung wird über das *SessionIf*-Objekt gesteuert. Insbesondere werden in der Session die Parameter einer Messung ausgetauscht, auf der Server-Seite wird darin der Empfang der Datenpakete realisiert. Nach Abschluss einer Messung werden die Messergebnisse an den Client zurückgeliefert. Der zeitliche Ablauf einer Messung wird in Kapitel 4.3.4 dargestellt.

4.2.1 Blocking-Mode

Prinzipiell können in einem *LTest*-Server beliebig viele Sessions gestartet werden. Dabei läuft jede Session in einem eigenen Thread ab, wodurch jede Session unabhängig von den anderen die Messdaten empfangen und verarbeiten kann. Einzelne Threads, in welchen die Sessions ablaufen, konkurrieren aber um die im System verfügbaren Prozessoren. Dadurch erhöht sich mit steigender Auslastung des Systems der maximale Fehler $\tau_{sched,max}$ (siehe Kapitel 2.5.2) sowie ggf. die Verzögerung der Datenpakete, verursacht durch Kollisionen und Stauungen im Netz. Um eine Möglichkeit zu schaffen, durch konkurrierende Mess-Datenströme verursachte Fehler zu vermeiden, wurde ein so genannter *Blocking-Mode* in *LTest* implementiert.

Wird vom Client eine Messung im Blocking-Mode angefragt, so wird im *FlowIf* überprüft, ob bereits weitere Sessions auf dem Server ablaufen. Sind keine Sessions auf dem Server vorhanden, wird eine neue blockierende Session erzeugt. Existieren bereits Mess-Sessions auf dem Server, so wird die Anfrage mit einer *BlockingException* abgelehnt. Ab dem Zeitpunkt, an dem eine blockierende Session im Server erzeugt worden ist, werden neue Client-Anfragen ebenfalls mit einer *BlockingException* abgelehnt. Ist die Ende-Zeit der blockierenden Session bekannt, so wird diese als Parameter der Exception an den Client zurückgegeben.

4.2.2 Messen zwischen zwei entfernten LTest-Servern

Mit dem in vorangehenden Abschnitten dargestellten Kommunikationsmodell ist lediglich eine gerichtete Messung der Netzwerk-Performance vom *LTest*-Client zum *LTest*-Server möglich. Bei Messungen im LAN und in Labor-Umgebungen ist dieses Kommunikationsmodell in vielen Fällen ausreichend. Wird dagegen ein Szenario betrachtet, in welchem das Messen beispielsweise als Service für Kunden im WAN angeboten wird, so wird der *LTest*-Server fest bei dem Service-Provider installiert. Die Kunden können sich dann zum *LTest*-Server des Providers verbinden und eine Mess-Session initiieren. Damit kann aber lediglich eine Messung der Netzwerk-Performance in der Richtung vom Kunden zum Anbieter realisiert werden. Um diese Einschränkung zu beseitigen, wurde das Kommunikationsmodell um einen so genannten *Remote Sender Mode* erweitert:

Das allgemeine Session-Interface *SessionIf* ist um zwei weitere Interfaces erweitert wor-

den – das *ReceiverSessionIf* und das *SenderSessionIf*. Das erste stellt eine Schnittstelle zur Empfänger-Implementierung der Session, das zweite die Schnittstelle zur Sender-Implementierung. Da die beiden Interfaces eine gemeinsame Basis haben (siehe Abbildung 4.3), kann das FlowIf-Objekt des *LTest*-Servers wahlweise einen Datensender oder einen Datenempfänger erzeugen.

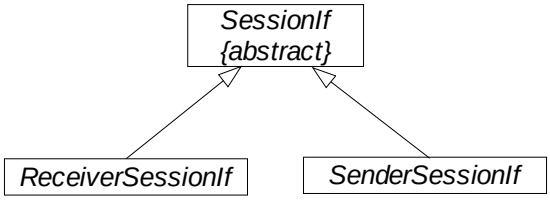

Abbildung 4.3: Ableitungsbaum der Session-Interfaces

Wird in einem *LTest*-Server ein Datensender instantiiert, so wird ein weiterer *LTest*-Server benötigt, welcher als Datenempfänger agiert. Die aktive Messung wird dann zwischen den beiden *LTest*-Servern durchgeführt (siehe Abbildung 4.4). Die Messergebnisse

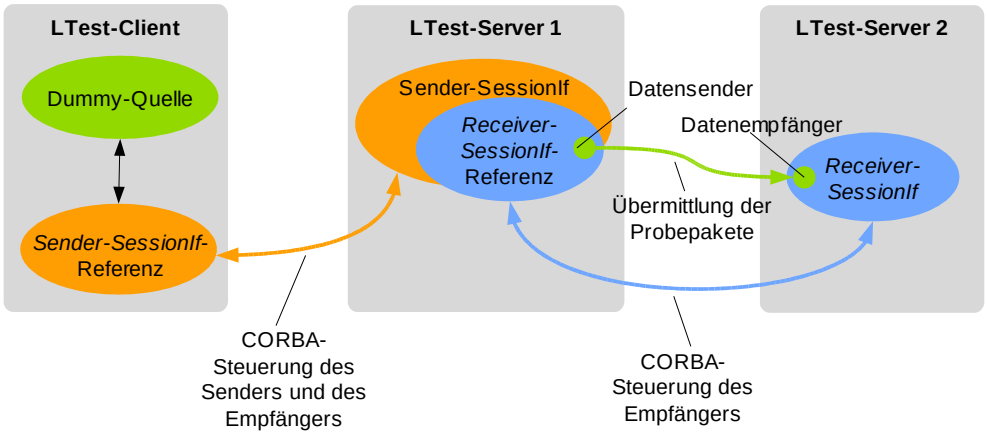

Abbildung 4.4: Kommunikation im Remote-Sender-Mode

werden nach Abschluss der Messung vom Datenempfänger zum Datensender im *LTest*-Server und von diesem anschließend zum *LTest*-Client übermittelt. Wird ein *LTest*-Server auf jenem Rechner gestartet, auf dem auch der *LTest*-Client läuft, so kann eine Messung vom Provider zum Nutzer durchgeführt werden (siehe Abbildung 4.5). Dabei stören sich Server und Client auf demselben System nicht, weil der Client während der Datenübertragung inaktiv ist (siehe Kapitel 4.3.4).

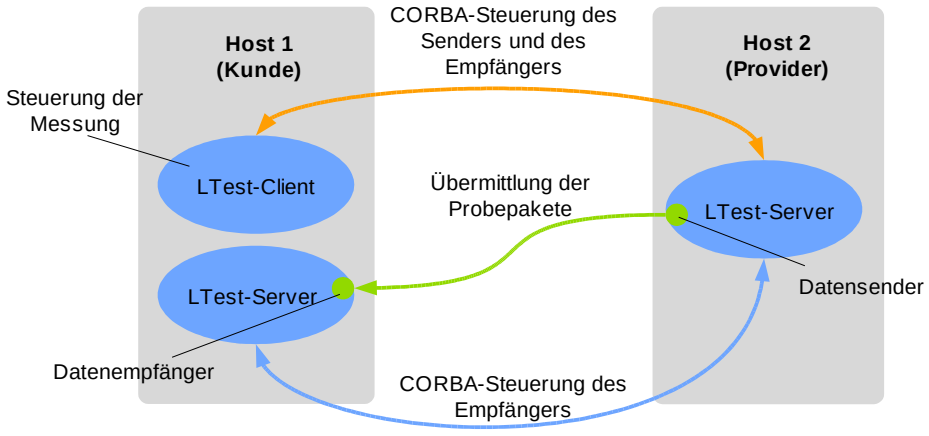

Abbildung 4.5: Messen zum lokalen Rechner

4.2.3 CORBA-Interface

Nachfolgend werden in Tabellen 4.1 und 4.2 die über CORBA realisierten Methoden und Exceptions vorgestellt.

Methode	Kurzbeschreibung der Methode
	FlowIf
start_rsession	Startet eine Empfänger-Session. Als Parameter wird die Quellenbeschreibung an den Empfänger übermittelt.
start_ssession	Startet eine Sender-Session. Als Parameter werden die Parameter der Quelle sowie die Quellenbeschreibung übergeben.
	SessionIf
release_session	Beenden des Session-Threads und Freigabe der Ressourcen der Session.
Port (Attribut)	Port-Adresse des Datenempfängers in der Receiver-Session.
	ReceiverSessionIf
finish_session	Der Datensender hat das letzte Datenpaket gesendet. Einleitung abschließender Maßnahmen. Als Parameter wird die Sequenznummer des letzten gesendeten Datenpakets übergeben.
	Fortsetzung auf der nächsten Seite

	Fortsetzung (siehe vorherige Seite)
get_samples	Übermittlung der Messergebnisse als Rückgabewert an den Datensender.
	SenderSessionIf
run_sender	Der Datensender wird auf dem *LTest*-Server gestartet. Als Parameter werden die Sender-Parameter sowie eine Senderbeschreibung für den Empfänger übergeben.
get_samples	Übermittlung der Messergebnisse als Rückgabewert an den *LTest*-Client.
getBasicSourceStats	Übermittlung der Messstatistiken an den *LTest*-Client.

Tabelle 4.1: Methoden der Client-Server-Kommunikation in *LTest*

Exception	**Kurzbeschreibung**
BlockingException	Blockieren bzw. Annehmen neuer Sessions nicht möglich. Attribute der Exception sind die Anzahl der gegenwärtig auf dem Server gehosteter Sessions sowie der Zeitpunkt, ab welchem der Server voraussichtlich frei wird.
NetworkException	Ein Kommunikationsfehler zwischen dem Datensender und dem Datenempfänger ist aufgetreten. Attribute der Exception sind der Fehlercode als Integer-Zahl und Fehlertext als String.

Tabelle 4.2: Exceptions des Interfaces *FlowIf*

4.3 Messvorgang

Nachdem die Session etabliert ist und die Parameter der Messung zwischen dem sendenden und dem empfangenden Teil der Session ausgehandelt sind, wird im *LTest*-Client bzw. in der Sender-Session, falls der Remote Sender Mode aktiviert ist, eine Datenquelle erzeugt, welche die Generierung und das Aussenden der Probepakete übernimmt.

4.3.1 Übermittlung der Datenpakete

Die übermittelten Datenpakete müssen alle für die Auswertung notwendigen Informationen enthalten und doch möglichst kompakt sein, um eine Modellierung von Datenströmen mit geringen Paketlängen zu ermöglichen. Eine der kleinsten PDU-Längen

wird bei einer Sprachübertragung nach G.729 verwendet (siehe Abbildung 2.2). Die in UDP eingekapselte RTP-PDU ist dabei 22 Byte lang. Soll in der Messung eine Sprachübertragung nach G.729 nachgebildet werden, so sollte die *LTest*-PDU diese Länge nicht überschreiten. Die *LTest*-PDU besteht grundsätzlich aus einem *LTest*-Header und einer Payload (siehe Abbildung 4.6). Der Header beinhaltet Informationen, die der Daten-

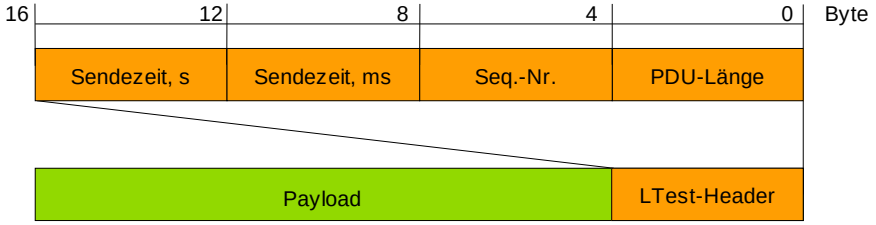

Abbildung 4.6: Format der *LTest*-PDU

empfänger verarbeiten soll. Die Payload wird mit einem zufälligen Bitmuster gefüllt.

Im *LTest*-Header müssen folgende Daten an den Empfänger übermittelt werden:

- **Sequenznummer des Datenpakets**. Diese wird benötigt, um die Empfangs-Zeitstempel den Sende-Zeitstempeln zuzuordnen sowie zur Identifikation von Paketverlusten und Packet-Reorderings. Die Sequenznummer wird als eine vorzeichenlose 32-Bit-Integer-Zahl codiert. Damit können rund 4 Mrd. Datenpakete ohne Zähler-Überlauf übertragen werden.

- **Paketlänge**. Diese wird für die Durchführung von Messungen über TCP benötigt. Die Angabe der Länge wird benötigt, um beim Empfänger *LTest*-PDUs aus dem vom TCP-Stack gelesenen Bytestrom zusammenzusetzen. Diese Aufgabe wird vom Reassembler des TCP-Empfängers übernommen (siehe Kapitel 4.3.3).

  Die PDU-Länge wird ebenfalls als eine 32-Bit-Integer-Zahl codiert. Die Länge von 32 Bit wurde zum einen aus Performance-Gründen gewählt, da die betrachteten Rechnersysteme mit 32-Bit-Integer-Zahlen schneller operieren als mit 16-Bit-Zahlen. Zum anderen sind TCP-Messungen denkbar, bei welchen es erwünscht ist, eine *LTest*-PDU in mehreren Datenpaketen zu übertragen. Damit wäre unter Umständen eine Codierung der PDU-Länge mit 16 Bit nicht ausreichend.

- **Sende-Zeitstempel**. Dieser muss mit der gewählten Kommunikationsstruktur nicht zwingend mitübertragen werden, da die Auswertung der Ergebnisse im Client erfolgt. In zukünftigen Szenarien ist es aber denkbar, dass auf der Server-Seite die Messergebnisse ebenfalls ausgewertet werden. Um auf solche Erweiterungen vorbereitet zu sein, ist der Sende-Zeitstempel ebenfalls in das Format der PDU

aufgenommen worden. Der Zeitstempel wird als zwei 32-Bit-Integer-Zahlen gespeichert. Die erste beinhaltet die Unix-Zeit in Sekunden, die zweite die Mikrosekunden.

Mit den oben genannten Feldern der *LTest*-PDU ergibt sich eine minimale Länge der PDU von 16 Bytes. Alle Integer-Zahlen werden mit dem *Most Significant Byte (MSB)* an erster Stelle serialisiert.

4.3.2 Implementierung der Datenquellen

Die Datenquelle wird direkt nach dem Start der Empfänger-Session im *LTest*-Client bzw. in der Sender-Session des *LTest*-Servers erzeugt. Die Aufgabe der Datenquelle ist es, nach einem vorgegebenen Algorithmus Zeitpunkte für das Aussenden der Datenpakete sowie deren Länge zu bestimmen und zu diesen Zeiten die *LTest*-PDUs auf den zugehörigen Socket zu schreiben.

Die Funktion der Datenquelle kann in folgende drei Phasen eingeteilt werden:

1. In der Initialisierungsphase werden die invarianten Parameter der Datenquelle berechnet. Soll die Messung zu einem Zeitpunkt gestartet werden, der mehr als 30 s in der Zukunft liegt, so wird die Datenquelle mit dem Systemaufruf *sleep()* suspendiert und erst 30 s vor dem Start der Übertragung der Messdaten reaktiviert. Mit dieser Maßnahme wird erreicht, dass *LTest*-Instanzen „auf Vorrat“ gestartet werden können, wobei durch die Angabe eines späteren Start-Zeitpunktes der Messung einzelne Instanzen einander nicht stören (siehe Kapitel 3.4.2).

   Des Weiteren wird vor Beginn einer Messung ein Socket mit den in der Receiver-Session ausgehandelten Parametern geöffnet. Ebenso wird im Konstruktor ein Buffer mit einem zufälligen Bitmuster erzeugt. Aus diesem wird während der Messung die Payload der *LTest*-PDU gefüllt. Die Verwendung eines zufälligen Bitmusters in der Payload der *LTest*-PDU verhindert eine Verzerrung der Messergebnisse, falls auf Teilstrecken eine Kompression der Daten erfolgt.

2. Die Ausführungsphase ist in der Methode *Dispatch()* realisiert. In dieser werden die Zeitpunkte für das Aussenden der Datenpakete sowie die zugehörigen Paketlängen bestimmt, *LTest*-PDUs gebildet und diese auf den Socket geschrieben.

   Da die Algorithmen für die Bestimmung der Zeitpunkte und der Paketlänge von dem gewählten Quellenmodell abhängig sind, ist *Dispatch()* in der Basisklasse als eine rein virtuelle Methode definiert. Diese Methode wird für jedes Quellenmodell einzeln implementiert. Die Funktionsweise dieser Methode ist im nachfolgenden Kapitel genauer erläutert.

3. In der Nachbearbeitungsphase wird der Socket für das Senden der Datenpakete geschlossen. Die in der Receiver-Session gespeicherten Ergebnisse werden an den Datensender übermittelt und dort den ausgesendeten Datenpaketen zugeordnet und gespeichert. Liegt eine Session mit einem entfernten Sender vor, so wird die Information über die ausgesendeten und empfangenen Datenpakete an die Dummy-Quelle des *LTest*-Clients weitergereicht. Anschließend werden grundlegende Statistiken über den Messvorgang gebildet. Ist in der Konfiguration die Speicherung der Ergebnisse einzelner Datenpakete vorgegeben, so werden diese in eine Datei geschrieben.

Um den Fehler der Messung zu minimieren, ist auf alle nicht zwingend notwendigen Operationen während der Datenübertragung verzichtet worden. So wird der *LTest*-Header, nachdem er erzeugt wurde, in eine Datenstruktur innerhalb der Datenquelle in einem einzelnen Kopiervorgang gespeichert. Das Schreiben der Daten auf eine Festplatte sowie die Bildung rudimentärer Statistiken wird erst nach Abschluss der Datenübertragung durchgeführt.

Da die Speicherung der Daten und die Bildung der Statistiken in der Datenquelle des *LTest*-Clients erfolgt, ist die Datenquelle im Client außerhalb der Session realisiert, damit die Session und damit auch der CORBA-Kanal unmittelbar nach dem Abschluss der Datenübertragung beendet werden kann. Dies ist besonders bei blockierenden Sessions relevant.

Das Interface der Datenquelle ist als abstrakte Klasse *CDataSource* realisiert. Von dieser werden unterschiedliche Datenquellen abgeleitet. Der Ableitungsbaum der Datenquellen ist in Abbildung 4.7 dargestellt.

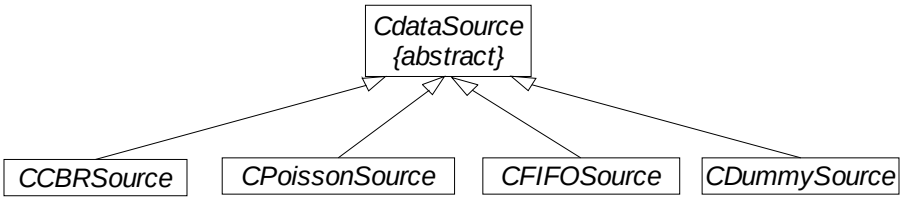

Abbildung 4.7: Ableitungsbaum der Datenquellen

Im Wesentlichen unterscheiden sich einzelne Datenquellen in der Berechnung der Zeitpunkte für das Aussenden der Datenpakete sowie der Paketlängen. Die bereits implementierten Datenquellen werden in den nachfolgenden Kapiteln vorgestellt. Neue Datenquellen können entweder in C++ durch eine Ableitung von *CDataSource* implementiert oder über das generische FIFO-Interface von einem externen Prozess an *LTest* zugeschaltet werden. Im ersten Fall muss die *Dispatch*-Methode der Klasse neu

implementiert werden, im zweiten Fall sind keine Änderungen am Quellcode von *LTest* notwendig (siehe Kapitel 4.3.2.4).

4.3.2.1 Packet-Scheduling in der Datenquelle

Die Realisierung des Packet-Scheduling im Datensender nimmt maßgeblich Einfluss auf die Güte des Messsystems. Wird die Aussendezeit für ein Datenpaket berechnet, so ist sicherzustellen, dass diese so genau wie möglich eingehalten wird. Die Güte der Einhaltung der Aussendezeit wird im Wesentlichen durch das in Kapitel 3.2.2 vorgestellte Process-Scheduling des Betriebssystems beeinflusst.

Das Packet-Scheduling wird in der Anwendungsschicht realisiert. Das Quantum für das Prozess-Scheduling beträgt in der Standard-Einstellung mindestens 10 ms [Ets03]. Wird zwischen dem Aussenden von Datenpaketen die CPU vom Sender-Prozess freigegeben, so kann die Aussendezeit eines Datenpakets im Worst-Case um mehr als 10 ms verfehlt werden.

Die Unsicherheit der Einhaltung der Wartezeiten bei der Freigabe der CPU wird in Anhang C vorgestellt. Um diese Unsicherheit zu minimieren, wird das Scheduling des Datensenders in einer *Busy-Wait-Schleife* realisiert [Tan01, S. 29 ff.]. Dafür wird die Methode *Dispatch()* der Datenquelle in einer Schleife aufgerufen und im jeden Schleifendurchlauf überprüft, ob die Aussendezeit erreicht ist. Wird sie erreicht, so wird das Datenpaket ausgesendet. Der wesentliche Nachteil des Busy-Wait-Schedulers ist die volle CPU-Auslastung durch den Prozess im Leerlauf[1] der Datenquelle. Damit sind aber sehr geringe Abweichungen der Aussendezeiten von den vorgegebenen Soll-Zeiten erreichbar. Ein Flussdiagramm der Methode *Dispatch* ist in Abbildung 4.8 dargestellt.

Ein verteiltes $n : m$-Messszenario kann realisiert werden, indem mehrere Mess-Clients simultan gestartet werden. Wegen der Verwendung des Busy-Wait-Schedulers der Datenquelle ist ein simultaner Lauf mehrerer Mess-Clients auf demselben Rechner nur sinnvoll, wenn folgendes gegeben ist:

- Es wird lediglich eine grobe Abschätzung der Netz-Parameter benötigt. Wegen der Konkurrenz mehrerer Busy-Wait-Scheduler um die CPU sind teilweise starke Abweichungen des Senders vom vorgegebenen Quellenmodell zu erwarten. Außerdem ist ein Anstieg des Fehlers τ_{sched} zu erwarten.
- Der Client wird auf einem Mehrprozessor-System gestartet. Hierbei soll nicht mehr als ein Mess-Client pro verfügbare CPU gestartet werden.

[1] Mit steigender Paketrate nimmt die CPU-Auslastung etwas ab wegen der Wartezeiten des Prozesses beim Aussenden der Datenpakete.

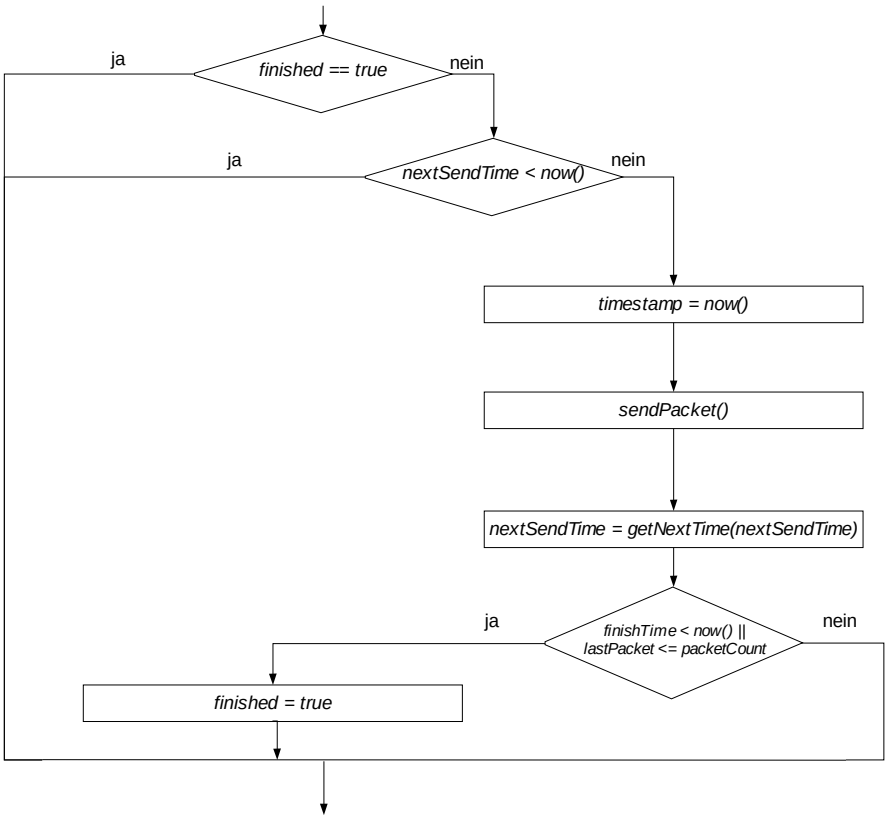

Abbildung 4.8: Flussdiagramm der Dispatch-Routine

4.3.2.2 CBR-Quelle

Die CBR-Quelle ist eine einfache und sehr häufig bei aktiven Messungen verwendete Datenquelle. Diese wird durch die Datenrate r in Bit/s und die Paketlänge S in $Byte$ charakterisiert. Bei der Initialisierung wird aus den oben genannten Parametern sowie aus der angegebenen Start-Zeit T der Zeitpunkt t_1 für das Aussenden des ersten Datenpakets nach folgender Vorschrift berechnet: $t_1 = max\{T, now\}$, wobei now die aktuelle Zeit ist. Da der zeitliche Abstand zwischen weiteren Datenpaketen δ_t konstant ist, kann dieser ebenfalls im Voraus berechnet werden:

$$\delta_t = \frac{8 \cdot S}{r} \tag{4.1}$$

Nach der Berechnung der Start-Zeit werden in einer Busy-Wait-Schleife die Datenpakete gesendet und anschließend die Aussendezeiten nachfolgender Pakete wie folgt berechnet:

$$t_{i+1} = t_i + \delta_t, \quad i > 0 \tag{4.2}$$

Ist die vorgegebene Anzahl Datenpakete gesendet worden oder ist der vorgegebene Zeitpunkt für das Beenden des Messvorgangs erreicht, so wird die Sende-Schleife verlassen.

4.3.2.3 Poisson-Quelle

Um das Problem der Synchronisierung bei der Realisierung verteilter Mess-Umgebungen mit einer großen Anzahl simultaner Mess-Datenströme zu vermeiden, wird in [Pax98a] empfohlen, in solchen Szenarien ein Poisson-Quellenmodell zu verwenden. In demselben Dokument sind auch die Details zur Berechnung der Aussendezeiten beschrieben.

Eine derartige Quelle ist ebenfalls als C++-Klasse in *LTest* implementiert. Diese wird durch dieselben Parameter wie eine CBR-Quelle beschrieben. Im Unterschied zur CBR-Quelle sind bei dieser Quelle die Zwischenankunftszeiten der Datenpakete exponentiell verteilt. Dafür wird die mittlere Paketrate R aus den Parametern Datenrate r und Paketlänge S nach der Berechnungsvorschrift (4.3) bestimmt:

$$R = \frac{r}{8 \cdot S} \tag{4.3}$$

Nachdem der Zeitpunkt für das Aussenden des ersten Datenpakets wie bei der CBR-Quelle berechnet worden ist, wird der zeitliche Abstand zum nachfolgenden Datenpaket nach folgendem Algorithmus bestimmt:

Es wird eine Zahl U_i von einem Zufallsgenerator mit einer Gleichverteilung der Zufallszahlen zwischen 0 und 1 erzeugt. Aus dieser wird anschließend der Momentanwert der Zwischenankunftszeit δ_i mit $\delta_i = -ln(\frac{U_i}{R})$ berechnet. Der Zeitpunkt für das Aussenden des nächsten Datenpakets ist $t_i + i = t_i + \delta_i, \quad i > 0$. Für das Erzeugen der Zufallszahl U_n wird die Funktion *rand()* [Ran00] der Standard-C-Bibliothek verwendet.

4.3.2.4 Generische FIFO-Quelle

Für die Realisierung diverser Datenquellen ohne Veränderung des Programmcodes von *LTest* wurde eine generische Quelle implementiert. Dabei wird erwartet, dass die Zeitpunkte für das Aussenden der Datenpakete sowie die Paketlängen von einem externen Programm berechnet werden. Als Schnittstelle zwischen der externen Datenquelle und *LTest* dient ein *FIFO-File*, auch als *named pipe* bezeichnet [Ste92, Kap.14.5]. Für jedes zu sendende Datenpaket wird eine durch ein Zeilenendzeichen (Hex-Code *0A*) abgeschlossene Zeile in das FIFO-File geschrieben. Diese muss wie folgt aufgebaut sein:

```
<SendingTime,seconds>„|" <SendingTime,microseconds>„|"<PacketLength>„\n"
```

Die vorgestellten Token werden als einfache Zeichenketten codiert. Die Bedeutung einzelner Token ist der Tabelle 4.3 zu entnehmen. Der Name des verwendeten FIFO-Files

kann als Kommandozeilenparameter oder über die Konfigurationsdatei angegeben werden.

Token	Bedeutung
`<SendingTime, seconds>`	Aussendezeit in Sekunden der Unix-Time
`<SendingTime, microseconds>`	Der Mikrosekunden-Anteil der Aussendezeit
`<PaketLength>`	Länge der *LTest*-PDU in Byte

Tabelle 4.3: Token des FIFO-Interfaces

Bei der Initialisierung der FIFO-Datenquelle wird ein zusätzlicher Programm-Thread gestartet, welcher das FIFO-File öffnet und blockierend daraus liest. Dieser Thread wird nachfolgend *Reader* genannt. Nach dem Parsen einer Zeile wird eine Sendemarke in eine interne Warteschlange aufgenommen, welche als Schnittstelle zwischen dem *Reader* und dem Datensender dient. Diese Warteschlange ist virtuell unendlich groß, so dass sie Sendemarken „auf Vorrat“ aufnehmen kann.

Beim Start des Datensenders sowie nach jedem Aussenden eines Datenpakets wird die Warteschlange auf Sendemarken überprüft. Wird eine Sendemarke gelesen, so wird entsprechend dieser Marke die Aussendezeit für das nächste Datenpaket gesetzt. Liegt die Aussendezeit bereits in der Vergangenheit, so wird das zugehörige Datenpaket sofort gesendet. Liegt die Zeit in der Zukunft, so wird in der *Dispatch*-Schleife auf den Zeitpunkt für das Aussenden des nächsten Datenpakets gewartet.

Ein gewisses Problem stellt das Beenden der FIFO-Datenquelle dar. Programmtechnisch ist die externe Quelle von dem Sender in *LTest* entkoppelt. Damit stellt sich die Frage, wie der *LTest*-Sender das Ende einer Session feststellen soll. Hierfür ist auf der einen Seite die Möglichkeit geschaffen worden, Parameter für die Start-Zeit, die Ende-Zeit sowie die Dauer einer Messung beim Programmstart anzugeben. In diesem Fall wird nach Ablauf der vorgegebenen Zeit das Senden beendet und die Verbindung zur FIFO geschlossen. Daraufhin schickt das Betriebssystem ein Signal an die externe Datenquelle, wodurch diese beendet wird. Außerdem kann, ebenso wie bei der CBR-Quelle, die maximal zu sendende Datenmenge angegeben werden. Beim Erreichen der vorgegebenen Datenmenge beendet die Datenquelle ebenfalls das Senden und schließt die FIFO.

Häufig ist es aber vorteilhaft, wenn die an das FIFO-Interface angeschlossene Quelle den Zeitpunkt für das Beenden der Messung bestimmt. Eine Möglichkeit, explizit das Ende für das Senden der Daten zu signalisieren, ist hierfür nicht vorgesehen. Stattdessen wartet der Sender mit einem Exponential-BackOff-Verfahren auf Daten aus der FIFO. Sind beim Lesen aus der FIFO keine Daten vorhanden, so wird ein *EOF*-Flag im Input-Stream des *Readers* gesetzt. In diesem Fall wartet der *Reader* 10 ms lang auf neue

Daten. Sind nach Ablauf dieser Zeit immer noch keine Daten in der FIFO vorhanden, so verdoppelt sich die Wartezeit. Kommen nach sieben Wartezyklen keine Daten, so wird der *Reader* beendet und dies im Sender vermerkt. Wird davor eine Marke über die FIFO eingelesen, so wird der Zähler zurückgesetzt. Hat sich der *Reader* beendet, so wird nach Abarbeiten der Queue der Sender ebenfalls beendet. Anschließend erfolgt die Nachverarbeitung der Daten.

Eine Alternative zur Verwendung des Timers mit dem Exponential-BackOff-Verfahren wäre die Einführung einer EOF-Marke für das FIFO-Interface. Diese Methode würde aber die Performance des *Readers* bei hohen Paketraten reduzieren, da beim Lesen jedes Tokens eine Überprüfung notwendig wäre, ob es sich um eine EOF-Marke oder um ein gültiges Token handelt. Der Timer wird hingegen nur beim Blockieren des *Readers* angewendet, wenn der Prozess wegen eines IO-Befehls suspendiert wird, und beeinflusst somit die Performance des *Readers* bei gefüllter Queue nicht.

Mit dem realisierten Verfahren zur Entkopplung des *Readers* vom Datensender kann statt eines FIFO-Files eine reguläre Text-Datei verwendet werden. Wird eine Prozess-Priorisierung für *LTest* verwendet (siehe Kapitel 4.4.1), so erhält der *Reader* eine um eins geringere Priorität als der Sender.

Bei der Verwendung der FIFO-Quelle ist auf den Buffer zwischen dem *Reader*- und dem Sender-Thread zu achten. Da der Reader- und der Sender-Thread um die CPU konkurrieren, können Pausen beim Auslesen der FIFO auftreten. Um sicherzustellen, dass die vorgegebenen Zeitpunkte für das Versenden eines Datenpakets in der Zukunft liegen, sollen die Sendemarken mindestens 100 ms im Voraus in das FIFO-File geschrieben werden. Ist genügend RAM im System vorhanden, können auch alle Marken für eine Messung „auf Vorrat“ geschrieben werden. Mit einem Speicherbedarf von unter 50 Byte pro Marke können auf modernen Systemen mehrere Mio. Marken in die Warteschlange zwischen dem Reader- und dem Sender-Thread aufgenommen werden.

4.3.2.5 Dummy-Datenquelle

Die Dummy-Datenquelle hat die Rolle eines Platzhalters. Sie wird im *LTest*-Client im Remote-Sender-Mode verwendet und sendet keine Datenpakete. Diese dient dem Empfang der Ergebnisse vom *LTest*-Sender, der Berechnung der Statistiken sowie der Speicherung der Ergebnisse.

4.3.3 Implementierung des Datenempfängers

Bei der Messung wird ein Datenempfänger im *LTest*-Server instantiiert. Der Datenempfänger existiert als Objekt in jeder Receiver-Session. Die wesentliche Aufgabe bei

der Realisierung des Datenempfängers ist eine möglichst genaue Feststellung des Zeitpunktes des Empfangs von Datenpaketen sowie eine effiziente Speicherung des *LTest*-Headers.

Da das Lesen auf dem Socket blockierend erfolgt, sind keine besonderen Scheduling-Mechanismen für den Empfänger erforderlich. Werden Receiver-Sessions vom *FlowIf* zugelassen, so werden keine weiteren Mechanismen zur Prozess-Synchronisierung zwischen unterschiedlichen Sessions vorgenommen. Um Verzögerungen durch die Verarbeitung der empfangenen Datenpakete zu minimieren, wird während des Datenempfangs lediglich ein Kopieren des *LTest*-Headers in einen internen Speicher der Session durchgeführt. Eine Weiterverarbeitung der Daten findet im Empfänger zurzeit nicht statt.

Beendet der Datensender das Senden der Datenpakete, so wird in der Receiver-Session die Methode *finish_session()* aufgerufen. Als Parameter wird die Sequenznummer des letzten gesendeten Datenpakets übermittelt. Innerhalb dieser Methode wird unter Verwendung eines Timeouts von 3 s auf das letzte gesendete Datenpaket gewartet. Ist es eingetroffen, so wird der Socket des Datenempfängers beendet und aus der Methode zurückgekehrt.

Anschließend kann der *LTest*-Client bzw. die Sender-Session die Messergebnisse des Empfängers anfordern. Zurück wird nicht der volle *LTest*-Header übermittelt, sondern lediglich die Sequenz-Nummer sowie der zugehörige Empfangs-Zeitstempel. Der Empfangs-Zeitstempel hat dasselbe Format wie der Sende-Zeitstempel. Es müssen für jedes empfangene Probepaket 12 Byte zurück an den Client übermittelt werden. Im Anschluss an die Übermittlung der Ergebnisse kann die Receiver-Session mit der Methode *release_session()* beendet werden. War dies eine blockierende Session, so werden ab diesem Zeitpunkt neue Verbindungen vom *FlowIf* wieder angenommen.

4.3.4 Zeitlicher Ablauf der Messung

Nachfolgend wird der zeitliche Ablauf des gesamten Messvorgangs mit *LTest* zusammengefasst. Dieser ist in Abbildung 4.9 schematisch dargestellt.

Nach dem Start sucht der *LTest*-Client nach dem *FlowIf*-Objekt auf dem angegebenen CORBA-Server. Anschließend wird vom *LTest*-Client eine neue *Receiver-Session* über das *FlowIf* erzeugt. Ist der Start einer Session erfolgreich, so wird im *LTest*-Client eine neue Datenquelle erzeugt. Diese kann sofort oder zeitversetzt mit dem Senden der Daten beginnen.

Während der Datenübertragung werden in der zugehörigen Receiver-Session keine Aktivitäten über die CORBA-Verbindung durchgeführt, um die Messergebnisse durch zusätzliche Kommunikation nicht zu beeinträchtigen. Nachdem der Austausch der Probepakete beendet ist, werden die Ergebnisse über die CORBA-Verbindung vom

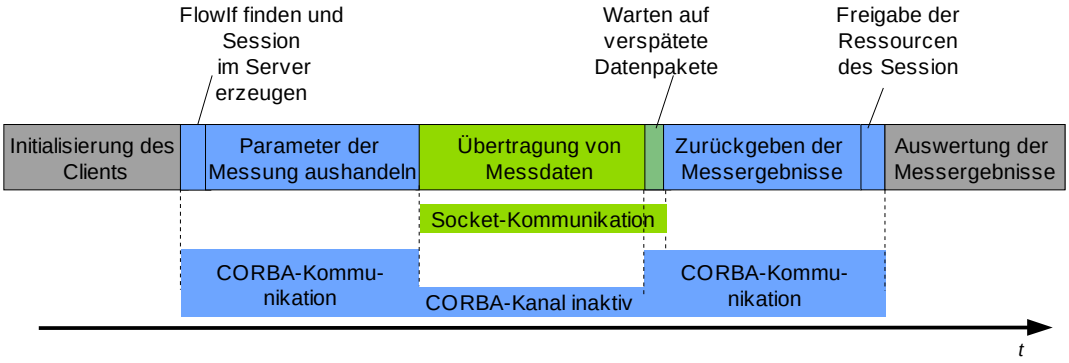

Abbildung 4.9: Kommunikationsphasen beim Messvorgang

Empfänger zum Datensender übermittelt. Danach wird die Receiver-Session beendet. Im Empfänger werden die mit dieser Session verbundenen Ressourcen freigegeben. Anschließend werden im *LTest*-Client Statistiken über die gesendeten und empfangenen Daten gebildet und bei Bedarf die Attribute einzelner Datenpakete in eine Datei geschrieben.

4.3.4.1 Besonderheiten beim Messen mit einem entfernten Sender

Das Messverfahren zwischen zwei entfernten *LTest*-Servern wurde in Kapitel 4.2.2 vorgestellt. Nachfolgend werden die Besonderheiten im zeitlichen Ablauf bei der Verwendung des Remote Sender Mode beschrieben.

Es wird zwischen dem *LTest*-Client und dem *LTest*-Server eine Sender-Session erzeugt. Dabei startet der *LTest*-Client diese Session und übergibt an den *LTest*-Server die Parameter der zu startenden Messung. Daraufhin wird auf der Server-Seite der Session eine Datenquelle erzeugt und eine Receiver-Session zwischen den beteiligten *LTest*-Servern initiiert (siehe Abbildung 4.4).

Ab diesem Punkt entspricht der Ablauf zwischen den beiden Servern bis auf das Zurückliefern der Messergebnisse dem Ablauf zwischen dem *LTest*-Client und dem *LTest*-Server im normalen Betriebsmodus. Nachdem die Ergebnisse beim Datensender gespeichert worden sind, werden diese weder ausgewertet noch ausgegeben. Vielmehr instantiiert der *LTest*-Client eine Dummy-Datenquelle (siehe Kapitel 4.3.2.5) und empfängt in dieser die Messergebnisse vom Sender. Die Dummy-Quelle verarbeitet die empfangenen Ergebnisse genauso wie die anderen Datenquellen nach dem Empfang der Messergebnisse.

Jeder *LTest*-Server kann wahlweise als sendender oder auch als empfangender Server verwendet werden. Wegen der Verwendung eines Busy-Wait-Schedulers im Datensender

(siehe Kapitel 4.3.2.1), ist eine Realisierung mehrerer sendender Sessions gleichzeitig nicht sinnvoll. Aus diesem Grund wird der *LTest*-Server immer bei dem Start einer Sender-Session blockiert. Der empfangende Server kann im normalen Betriebsmodus für mehrere simultane Sessions verwendet werden.

4.4 Maßnahmen zur Verbesserung der Performance

Nachfolgend werden einzelne Maßnahmen zur Reduktion der Messfehler sowie für die Erhöhung der Performance von *LTest* vorgestellt.

4.4.1 Prozess-Priorisierung und das Blockieren des Arbeitsspeichers

Es kann eine Prozess-Priorisierung mit der Priorität zwischen 1 und 99 vorgenommen werden. Dabei wird auf der Sender-Seite der Prozess-Thread, in welchem der Datensender läuft, mit der angegebenen Priorität versehen. Auf der Empfängerseite wird lediglich der Thread, in dem der Empfänger läuft, priorisiert. Dabei wird als Scheduling-Verfahren das Round-Robin-Scheduling gewählt [Set02]. Außerdem wird mit dem Systemaufruf *mlockall* [Mlo96] die Auslagerung des Arbeitsspeichers auf einen Massenspeicher verhindert. Beide Maßnahmen sind auf Unix-Systemen nur für den Superuser verfügbar. Der Sender-Prozess und der Empfänger-Thread werden immer mit derselben Priorität versehen, eine unabhängige Priorisierung des Senders und Empfängers ist nicht vorgesehen.

4.4.2 Speicherung der Ergebnisse

Wie bereits in Kapitel 4.3.2 beschrieben, wird während der Übertragung der Probepakete auf Zugriffe auf langsame Hardware, wie z.B. Festplatten, wo immer möglich, verzichtet, um dadurch verursachte Verzögerungen zu vermeiden. Aus diesem Grund werden die Messergebnisse während einer Messung im Arbeitsspeicher gehalten.

Im Sender werden für jedes ausgesendete Datenpaket die Sequenznummer, die Aussendezeit und die Paketlänge gespeichert. Außerdem wird bereits beim Aussenden der Datenpakete im Sender Speicherplatz für den Empfangs-Zeitstempel reserviert. Damit wird verhindert, dass eine Ausschöpfung des Arbeitsspeichers bei der Übermittlung der Messergebnisse zurück zum Sender stattfindet.

Die oben genannten Informationen werden in einem einfachen Byte-String gespeichert, welcher denselben Aufbau hat wie die Bytes 0-15 der *LTest*-Headers (siehe Abbildung 4.6) zuzüglich des Empfangs-Zeitstempels im selben Format wie der Sende-Zeitstempel. Damit werden die Daten sehr effizient gespeichert, weil dafür nur ein

memcpy-Systemaufruf notwendig ist. Die Länge des Byte-Feldes für jedes Probepaket beträgt 24 Byte. Diese Byte-Strings werden in einer Vector-Datenstruktur der *C++-Standardbibliothek* [Str00, Kap. 16] abgelegt. Die Deklaration sieht folgendermaßen aus:

```
struct CPDUHeader{
 // format is:  Bytes 0-15: packet header like in PDU, 16-23 receiving
 // timestamp: seconds and microseconds in host byte order
 char Header [24];
...
 // constructor with initializing data to header of the pointed packet
 CPDUHeader(char * pPacket){
   memcpy(Header, pPacket, 16);
  }
};
...
class CDataSource {
...
vector <CPDUHeader> mHeaderVec;    // vector with source samples
...
};
```

Ist bei einer Datenquelle die Anzahl der zu sendenden Datenpakete während der Instantiierung bekannt, so wird für den Vector bereits im Konstruktor der benötigte Speicherplatz alloziert. Damit werden während der Datenübertragung Kopiervorgänge in Verbindung mit der Verwaltung der Datenstruktur weitgehend vermieden.

Beim Datenempfänger wird für jedes empfangene Datenpaket ein Byte-String der Länge 12 benötigt. Hier wird der Empfangs-Zeitstempel sowie die zugehörige Sequenznummer gespeichert. Da im Empfänger im Allgemeinen die Anzahl der in einer Session zu erwartenden Datenpakete unbekannt ist, ist eine effiziente Vorbelegung auf eine passende Größe nicht möglich. Hier wird für eine möglichst effiziente Speicherverwaltung eine Kombination aus einem Vektor von Arrays mit PDU-Headern verwendet. Damit wird ein Kompromiss zwischen einer einfachen Speicherverwaltung und sparsamer Nutzung des Speichers geschlossen.

4.5 Darstellung und Weiterverarbeitung der Ergebnisse

Die Darstellung der Ergebnisse wurde unter Berücksichtigung der in Kapitel 3.5 vorgestellten Gesichtspunkten implementiert. Eine Zusammenfassung der Messparameter sowie ein Satz rudimentärer Statistiken wird in Textform unmittelbar nach Abschluss

der Messung ausgewertet und vorgestellt. Nachfolgend wird beispielhaft das Ergebnis einer Messung dargestellt:

```
Session duration, seconds:      10.2
Packets sent: 3 064 788
Bytes sent: 76 619 700
Sending data rate, Bit/s: 6.12957e+07
Packets received: 3 064 788
Bytes received: 76 619 700
Receiving data rate, Bit/s: 6.12957e+07
Packets duplicated: 0
Corrupted packets:  0
Lost packets:       0
Packet loss:        0 %

============
Statistics:|
=========================================
Parameter  |   Packets   |    Bytes     |
=========================================
Sent       |  3064788    | 7.66197e+07  |
Received   |  3064788    | 7.66197e+07  |
Dropped    |        0    | 0            |
Loss (%)   |        0    | 0            |

=================================================================================
           |        Sent       |        Received    |        Delay      | Length
=================================================================================
Parameter |InterPacketTime [s]| InterPacketTime [s]|       [s]         | [bytes]
=================================================================================
Min        | 0000000000.000002 | 0000000000.000000  | 0000000000.008751 |    25
Seq.No.    |                51 |                 4  |                34 |  3064787
---------------------------------------------------------------------------------
Max        | 0000000000.128332 |  0000000000.109783 | 0000000000.136796 |    25
Seq.No.    |           2097153 |            2097152 |           2097152 |     0
---------------------------------------------------------------------------------
Mean       |          3.26e-06 |           3.26e-06 |           0.00989 |    25
---------------------------------------------------------------------------------
St. deviation |      8.91e-05 |           0.000102 |            0.0025 |     0
---------------------------------------------------------------------------------
```

Für eine genauere Analyse der Messergebnisse kann ein Trace – eine Sammlung der Parameter aller gesendeten und empfangenen Datenpakete – der Messung in eine Datei geschrieben werden. Dabei enthält jede Zeile die Information über ein Probepaket. Nachfolgend wird beispielhaft ein Auszug eines Trace gezeigt:

```
# Format of the trace file is:
```

```
# <Sequence number>\t
# <sentTime_sec>.<sentTime_microsec>\t
# <recvTime_sec>.<recvTime_microces>\t
# <Delay_sec>.<Delay_microsec>\t
# <Packet_length>\n
0   1088630463.828572  1088630463.822938   0000000000.005634      56
1   1088630463.869510  1088630463.863772   0000000000.005738      56
```

Dieser Trace dient der weiteren Verarbeitung bzw. der Analyse der Messergebnisse. Damit kann eine genaue Auswertung der Ergebnisse von *LTest* entkoppelt werden.

Wie bereits in Kapitel 2.6.4 dargestellt, sind die Verteilungen einzelner Größen für die Darstellung der Ergebnisse von Bedeutung. Für die Erzeugung solcher Verteilungen stehen zwei Programme zur Verfügung. Das erste ist das Perl-Script *ltest_stats.pl*. Mit diesem wird aus dem Trace ein Satz von Text-Dateien erzeugt, in welche in tabellarischer Form Rohdaten für das Zeichnen einer PDF sowie einer CDF bzw. CCDF der Messergebnisse geschrieben werden. Es wird jeweils eine Datei für die Verteilungen der Einweg-Verzögerungen, der Sende-Jitter, der Empfangs-Jitter sowie der Paketlängen geschrieben. Mit Hilfe dieser Dateien können die Verteilungsfunktionen z.B. mit *gnuplot* gezeichnet werden. Ein Export der mit *ltest_stats.pl* erzeugten Daten nach Microsoft Excel oder StarOffice ist ebenfalls möglich.

Mit der Entwicklung des Tools namens *LTestView* (siehe Kapitel 4.5) wurde das Ziel verfolgt, eine direkte Einbindung der Messergebnisse in Web-Seiten zu ermöglichen. Dabei soll die Flexibilität des SVG-Formats zum Tragen kommen. Mit *LTestView* werden aus der Trace-Datei CDFs und PDFs der gemessenen Größen im SVG-Format geschrieben. Diese können in einem SVG-fähigen Web-Browser dargestellt werden. Abbildung 4.10 zeigt eine Auswahl der von *LTestView* erzeugten Darstellungen der Messergebnisse. Mit SVG soll es laut Standard beispielsweise möglich sein, bestimmte Bereiche eines Bildes auszuwählen und dynamisch zu skalieren sowie den Wert eines Messpunktes bei einer Bewegung der Maus über diesen Punkt anzuzeigen. Diese Features müssen aber auch von den verwendeten Viewern – in erster Linie den Web-Browsern – unterstützt werden.

Tests im Rahmen dieser Arbeit haben aber gezeigt, dass die Unterstützung des SVG-Formats in aktuellen Web-Browsern noch nicht in ausreichendem Maße gegeben ist. In den Web-Browsern Mozilla 1.7 sowie im Internet Explorer 6.0 ist eine Skalierung von SVG-Bildern nicht möglich. Ebenso ist eine dynamische Anzeige der Werte einzelner Messpunkte beim Drüberfahren mit der Maus nicht möglich. Es haben sich auch einige Inkompatibilitäten in der Verarbeitung der SVG-Dateien zwischen Mozilla und Internet Explorer herausgestellt. Wegen der genannten Mängel der SVG-Implementierungen wird zurzeit nur eine statische Einbindung von SVG-Grafiken fester Größen unterstützt.

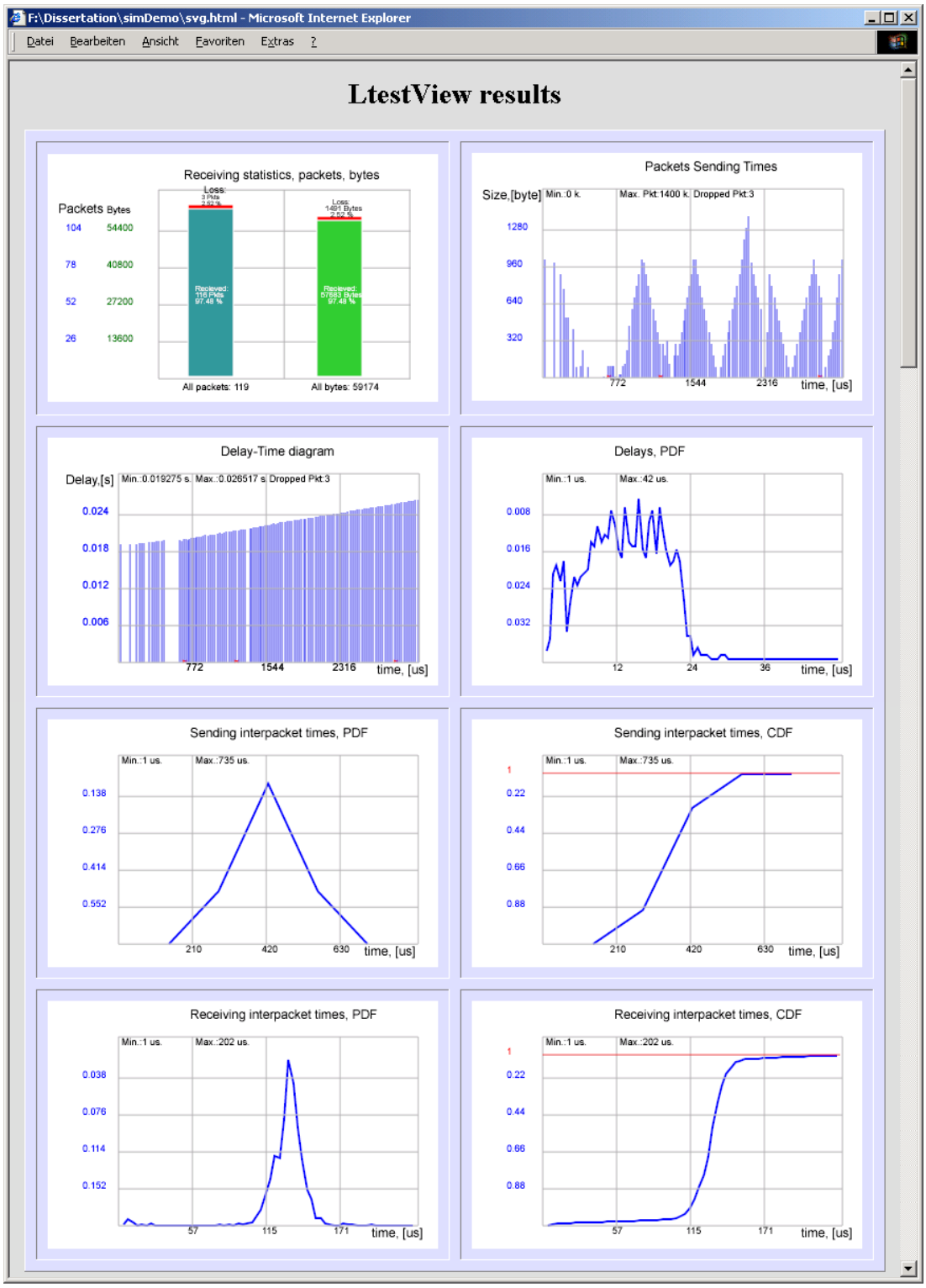

Abbildung 4.10: Darstellung der Ergebnisse im Web-Browser

Kapitel 5

Evaluierung des Systems sowie mögliche Messszenarien

Nachfolgend werden die in unterschiedlichen Messszenarien gewonnenen Ergebnisse vorgestellt, welche die Funktion sowie die Performance von *LTest* beschreiben. Außerdem wird eine Bewertung getroffener Maßnahmen zur Reduzierung der Messfehler vorgenommen. Anschließend werden typische Messszenarien für die Untersuchung der Dienstgüte und Netzwerk-Performance sowie die Auswirkung unterschiedlicher Parameter der Messung auf die erzielten Ergebnisse vorgestellt.

5.1 Güte des Systems

Bei der Evaluierung des Systems steht die Untersuchung der Funktion von *LTest* sowie die Wechselwirkung zwischen der Hardware, dem Betriebssystem und *LTest* im Vordergrund. Für diese Untersuchung wird eine Messumgebung gemäß Abbildung 5.1 aufgebaut. Beide beteiligten Systeme laufen unter Linux mit der Kernel-Version 2.4.21. In diesem Szenario werden unter anderem die Güte der Datenquellen und der Einfluss der Prozesspriorisierung untersucht sowie der Fehler τ_{sched} abgeschätzt.

Durch den Verzicht auf zwischengeschaltete aktive Komponenten wird die Anzahl der am Datentransport beteiligten Buffer und Vermittlungssysteme reduziert und damit der Einfluss des Netzes auf das System minimiert. Dadurch lassen sich die Ursachen für Störungen der Messungen mit *LTest* und Auffälligkeiten in den Messergebnissen einfacher interpretieren. Wenn nichts anderes erwähnt, wird die Uhrensynchronisation über zwei DZF77-Referenz-Uhren realisiert [Mei04, Grö04].

5.1.1 Datenquelle

Die wesentliche Aufgabe der Datenquelle ist das Aussenden von Datenpaketen vorgegebener Größe zu den durch das Quellenmodell festgelegten Zeitpunkten. Damit ergeben sich zwei Kriterien für die Güte der Datenquelle:

- die Abweichung der realen Aussendezeit von der errechneten. Hierfür bietet sich die Betrachtung der Sender-Jitter gemäß Kapitel 2.3.3 an.

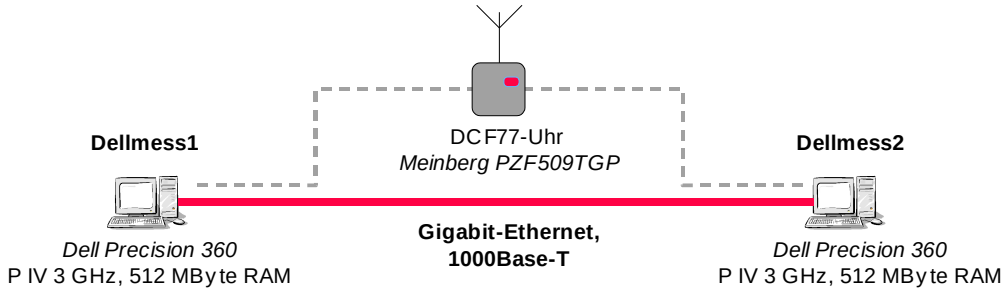

Abbildung 5.1: Messaufbau für eine Punkt-zu-Punkt-Messung

- die maximale mit *LTest* erreichbare Datenrate bzw. Paketrate sowie sonstige Parameter, welche die Performance des Systems quantifizieren. Diese Aspekte werden in Kapitel 5.3 genauer betrachtet.

Die allgemeinen Untersuchungen der Güte sind mit der CBR-Quelle durchgeführt worden. Die Verwendung einer CBR-Quelle hat den Vorteil, dass im Idealfall die Pakete äquidistant ausgesendet werden, d.h. die Standardabweichung der Zwischenankunftszeiten der Datenpakete soll gegen null gehen.

5.1.1.1 CBR-Quelle

Für die Untersuchung der Güte der Datenquelle sowie für die Bestimmung der Ursachen der dabei auftretenden Fehler wird in zwei Messreihen der Sender-Jitter in Abhängigkeit von der Paketrate sowie von der Datenrate betrachtet. Jede einzelne Messung einer Messreihe dauert 20 s, als Transportprotokoll wird UDP verwendet. Jeder Messpunkt entspricht dabei einer statistischen Auswertung einer solchen Messung mit einer bestimmten Paketlänge sowie einer vorgegebenen Sender-Datenrate.

In der ersten Messreihe wird bei einer konstanten Paketrate von 10.000 Pakete/s die Datenrate von 2,24 MBit/s schrittweise auf 120 MBit/s erhöht. Dies wird durch die Veränderung der *LTest*-PDU zwischen 28 und 1.500 Byte sowie der Übertragungsrate zwischen 2,24 und 120 MBit/s erreicht. Es werden 200.000 Datenpakete in jeder einzelnen Messung ausgesendet. Die Messergebnisse sind in den Abbildungen 5.2 und 5.3 dargestellt. Dabei bezieht sich die Standardabweichung auf den gemessenen Mittelwert. Aus dem Verlauf der Standardabweichung ist ersichtlich, dass die Güte der Datenquelle nahezu keine Abhängigkeit von der übertragenen Datenrate und somit von der Paketlänge aufweist. Dabei ist zu beachten, dass die Datenrate im letzten Messpunkt 54 mal größer ist als im ersten.

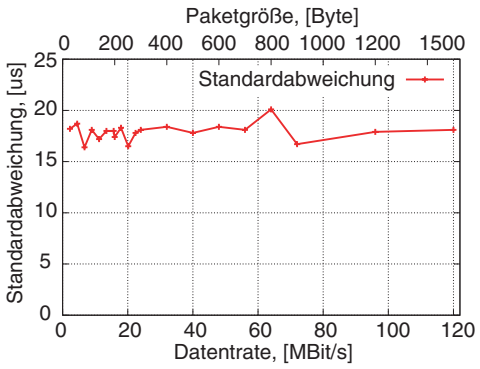

Abbildung 5.2: Standardabweichung der Zwischenankunftszeiten am Sender bei Veränderung der Datenrate

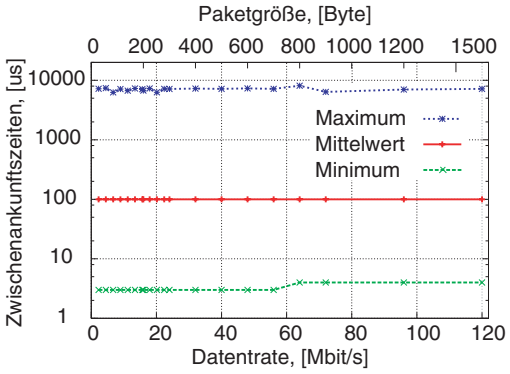

Abbildung 5.3: Mittlere, minimale und maximale Zwischenankunftszeiten am Sender bei Veränderung der Datenrate

Die dargestellten Werte für die Sender-Jitter charakterisieren nur die Güte der Einhaltung der Aussendezeiten vom Anwendungsprogramm. Ob eine zum vorgegebenen Zeitpunkt erzeugte PDU auch zeitnah auf die physische Leitung gegeben wird, ist an dieser Stelle noch offen. Diese Frage wird in Kapitel 5.1.4 beantwortet. Mit dem Erzeugen der PDU wird auch der Zeitstempel für das Aussenden des Datenpakets gesetzt. Treten Verzögerungen zwischen dem Erzeugen einer PDU und deren Aussenden auf, so wird die Laufzeit fehlerhaft erfasst. Dieser Fehler fließt in die in Kapitel 2.5.2 beschriebene Komponente τ_{sched} ein.

In einer weiteren Messreihe werden Messungen mit einer konstanten Datenrate von 2,24 MBit/s durchgeführt. Die Paketrate wird dabei schrittweise von 200 auf 10.000 Pakete/s erhöht. Die höchste Paketrate dieser Reihe entspricht damit der konstanten Paketrate aus der vorherigen Messung. Dabei wird die Veränderung der Paketrate durch die Variation der Paketlängen zwischen 1.400 und 28 Bytes erreicht. Die Standardabweichung sowie die mittleren, minimalen und maximalen Zwischenankunftszeiten am Sender sind in Abbildungen 5.4 und 5.5 dargestellt.

Ab einer Paketrate von 1.400 Pakete/s steigt die Standardabweichung stetig mit der Paketrate. Der Abbildung 5.5 kann entnommen werden, dass ebenso wie die Standardabweichung das Maximum der Zwischenankunftszeit mit steigender Paketrate ansteigt. Um das Verhalten der Datenquelle genauer zu beleuchten, wird in den Abbildungen 5.6 und 5.7 die CCDF der Verteilung der Zwischenankunftszeiten für den letzten Messpunkt dieser Messreihe dargestellt. Tabelle 5.1 fasst die relevanten Ergebnisse dieser Messung zusammen.

Der Abbildung 5.6 kann entnommen werden, dass drei Datenpakete aus 200.000 eine Zwischenankunftszeit mehr als eine Millisekunde haben. Um die mittlere Datenrate einzuhalten, werden zum Ausgleich für die zu großen Zwischenankunftszeiten mehrere

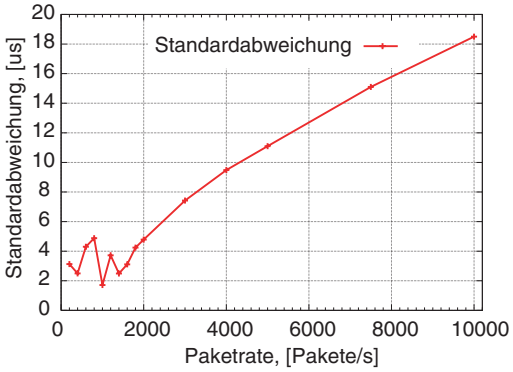

Abbildung 5.4: Standardabweichung der Zwischenankunftszeiten im Sender bei Veränderung der Paketrate

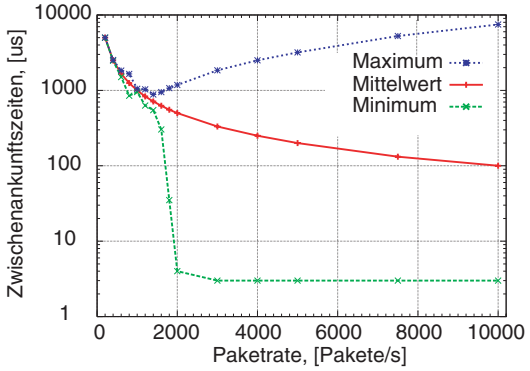

Abbildung 5.5: Mittlere, minimale und maximale Zwischenankunftszeiten im Sender bei Veränderung der Paketrate

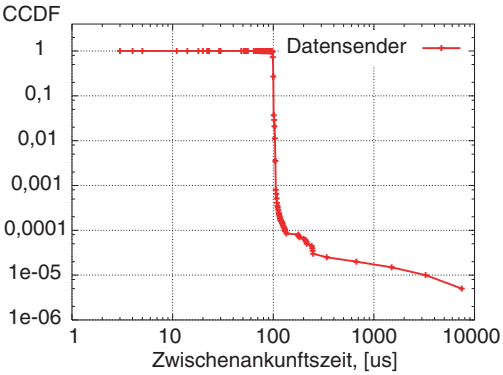

Abbildung 5.6: CCDF der Zwischenankunftszeiten an der Datenquelle bei einer Paketrate von 10.000 Pakete/s, log-log

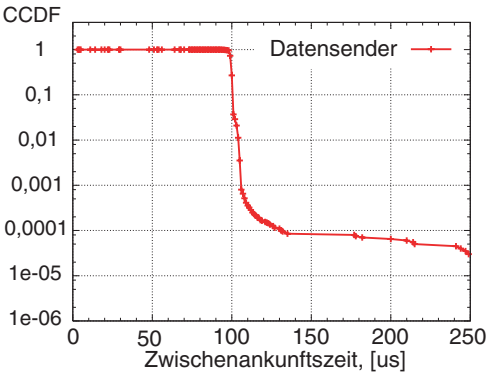

Abbildung 5.7: CCDF der Zwischenankunftszeiten an der Datenquelle bei einer Paketrate von 10.000 Pakete/s, lin-log

Datenpakete direkt nacheinander gesendet. Dabei ist der minimale Abstand durch die Serialisierungsverzögerung und die minimale Verarbeitungszeit der Datenpakete limitiert. Aus diesem Grund sinkt die Minimum-Kurve mit steigender Paketrate bzw. sinkender Paketlänge in die Nähe des minimalen Abstands zwischen Datenpaketen auf einer Gigabit-Ethernet-Verbindung von 3 µs (siehe Tabelle 2.4 auf Seite 30). In anderen Messpunkten verlaufen die entsprechenden CCDF ähnlich. Abbildung 5.8 stellt beispielsweise die CCDF für den Messpunkt 1.200 Pakete/s dar. Es wird lediglich eine geringere Streuung um den Mittelwert festgestellt, was in einer geringeren Auslastung des Systems begründet ist.

Aus der Darstellung der Minima der Zwischenankunftszeiten kann die minimale Verarbeitungszeit im System abgeschätzt werden. Diese entspricht den gemessenen Minima abzüglich der Serialisierungszeiten der Datenpakete. Damit dauert das Vorbereiten einer PDU und das Schreiben auf den Socket weniger als 1 µs.

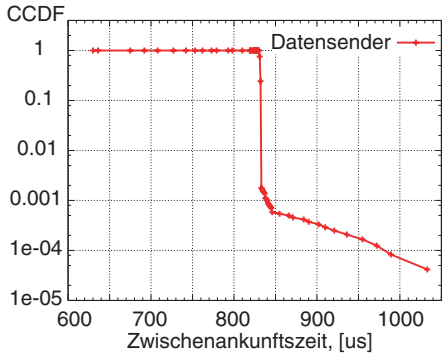

Abbildung 5.8: CCDF der Zwischenankunftszeiten an der Datenquelle bei einer Paketrate von 1.200 Pakete/s, lin-log

Größe	**Wert**
Dauer der Session [s]	20,0
Anzahl gesendeter Datenpakete	200.000
mittlere Datenrate des Senders [MBit/s]	2,4
Anzahl verlorener Datenpakete	10
Paketverlustrate [%]	0,05
mittlere Zwischenankunftszeit im Sender [µs]	100
minimale Zwischenankunftszeit im Sender [µs]	3
maximale Zwischenankunftszeit im Sender [µs]	7.471
Standardabweichung der Zwischenankunftszeiten am Sender [µs]	18, 5

Tabelle 5.1: Zusammenfassung der Messergebnisse im Messpunkt 10.000 Pakete/s, 28 Byte

Von Interesse ist in diesem Zusammenhang der Vergleich gemessener Ergebnisse der Zwischenankunftszeiten mit den mit der Tschebyscheffschen Ungleichung (siehe Kapitel 2.4.2.3) errechneten Grenzwerten. Am Beispiel der vorgestellten Messung mit 10.000 Pakete/s ist in Tabelle 5.2 ein solcher Vergleich für die Abweichung um 5, 10 und 50 % vom Mittelwert sowie die Wahrscheinlichkeit für das Auftreten des Wertes der gemessenen maximalen Verzögerung dargestellt. Als Γ wird in der Tabelle der Schwellwert der Abweichung der Zwischenankunftszeit vom Mittelwert m bezeichnet.

Für die dargestellte Messung wird die Ungleichung zwar formal bestätigt, liefert für geringe Schwellwerte Γ aber für die Praxis keine brauchbaren Ergebnisse (erste drei Zeilen der Tabelle). Für große Werte der Abweichungen liefert die Ungleichung eine relativ gute Abschätzung der Wahrscheinlichkeit. Diese Beobachtung ist bei anderen

Schwellwert Γ der Abweichung der Zwischenankunftszeit	$P(\|(t_{n+1} - t_n) - m\| \geq \Gamma)$ **gemessen**	**Grenzwert nach Tschebyscheff**
$\pm$ 5 % $\hat{=}$ $\pm$ 5 μs	$5,26 \cdot 10^{-3}$	> 1
$\pm$ 10 % $\hat{=}$ $\pm$ 10 μs	$1,35 \cdot 10^{-3}$	> 1
$\pm$ 50 % $\hat{=}$ $\pm$ 50 μs	$8 \cdot 10^{-4}$	$1,369 \cdot 10^{-1}$
+7.371 % $\hat{=}$ + 7.371 μs	$5 \cdot 10^{-6}$	$6,13 \cdot 10^{-6}$

Tabelle 5.2: Wahrscheinlichkeit der Überschreitung bestimmter Schwellwerte der Zwischenankunftszeiten

im Laufe der Arbeit durchgeführten Messungen ebenfalls gemacht worden. Wegen der zu hoch geschätzten Schwellwerte für geringe Abweichungen vom Mittelwert wird die Tschebyscheffsche Ungleichung in dieser Arbeit nicht weiter verwendet.

Des Weiteren kann festgestellt werden, dass bei Paketraten unter 1.400 Pakete/s die Standardabweichungen mit steigenden Paketraten keinen stetigen Anstieg aufweisen (siehe Abbildung 5.4). Um dieses Verhalten zu verdeutlichen, wird eine weitere Messreihe mit demselben Messaufbau durchgeführt. Die Paketrate wird dabei über einen größeren Wertebereich – von 10 bis 150.000 Pakete/s – verändert. Es werden Messreihen mit Paketlängen von 150 Byte und 1.500 Byte jeweils unter Verwendung von TCP und UDP durchgeführt. Abbildung 5.9 zeigt die Standardabweichungen der Sender-Jitter. Mit 1.500-Byte Datenpaketen ist nur der Bereich bis 85.000 Pakete/s relevant, weil damit bereits die Link-Datenrate von 1 Gbit/s überschritten wird. Abbildung 5.10 zeigt einen Ausschnitt der Ergebnisse mit Paketraten bis 3.000 Pakete/s. Dieser Wertebereich wird separat dargestellt, um die geringere Güte der Datenquelle bei geringen Paketraten hervorzuheben.

Die wesentlichen Ursachen für den Sender-Jitter sind zum einen die Unterbrechung des *LTest*-Prozesses durch den Scheduler des Betriebssystems und zum anderem das Blockieren des Prozesses bei dem Schreiben der *LTest*-PDUs, wenn der IP-Stack bereits gefüllt ist. Das Blockieren des Prozesses beim Schreiben einer PDU auf den Socket ist auf die Performance des IP-Stacks im Betriebssystem sowie auf den Treiber und die Netzwerk-Hardware zurückzuführen und kann von der Anwendung nicht beeinflusst werden. Der Anstieg der Sender-Jitter als Folge der Auslastung des IP-Stacks ist in den o.g. Abbildungen bei Paketlängen von 1.500 Byte im Bereich 70.000 – 85.000 Pakete/s erkennbar. Da dieser Fall nicht die Güte der Anwendung, sondern vielmehr die Performance des Betriebssystems und des Netzwerk-Treibers repräsentiert, wird dieser hier nicht weiter betrachtet.

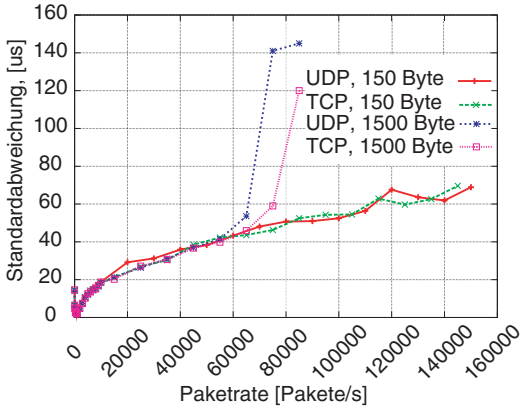

Abbildung 5.9: Standardabweichung der Sender-Jitter bei einer Paketrate von 10 – 150.000 Pakete/s

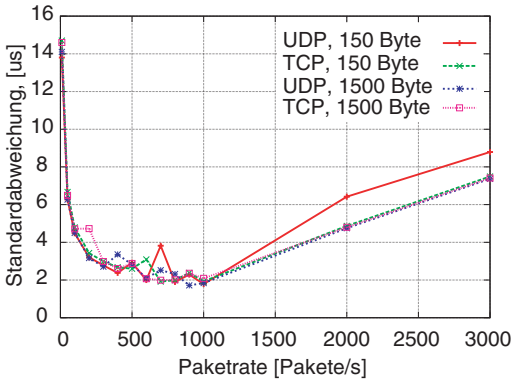

Abbildung 5.10: Standardabweichung der Sender-Jitter bei einer Paketrate von 10 – 3.000 Pakete/s

Die Auswirkung der Prozess-Unterbrechung auf die Sender-Jitter wird nachfolgend vorgestellt:

Wie bereits in Kapitel 4.3.2.1 beschrieben, wird für den Datensender ein Busy-Wait-Scheduler verwendet. Für das Versenden werden die *LTest*-PDUs auf den Socket geschrieben. Dabei wird der *LTest*-Prozess vom Betriebssystem suspendiert, damit in dieser Zeit andere Anwendungsprozesse oder Kernel-Threads die CPU-Zeit erhalten. Bei geringen Paketraten finden solche, durch das Schreiben auf den Socket bedingte Kontext-Switches sehr selten statt. Damit finden diese aber häufiger an zufälligen Zeitpunkten statt, wodurch die Wahrscheinlichkeit ansteigt, dass zum Zeitpunkt des Aussendens eines Datenpakets der *LTest*-Prozess suspendiert ist. Das Datenpaket wird dann vom *LTest-Prozess* verspätet, unmittelbar nach der erneuten Zuweisung der CPU, ausgesendet. Bei steigender Paketrate (Bereich <1.000 Pakete/s) steigt die Häufigkeit der Bedienung konkurrierender Prozesse direkt nach dem Aussenden der PDU. Die Bearbeitung konkurrierender Prozesse wird damit auf die Aussendezeiten der *LTest*-PDUs synchronisiert, wodurch die Güte der Datenquelle verbessert wird.

Bei weiter steigenden Paketraten (Bereich >1.000 Pakete/s in den betrachteten Abbildungen) steigt die Wahrscheinlichkeit dafür, dass die Verweildauer eines konkurrierenden Prozesses auf der CPU die vorgesehene Zwischenankunftszeit übertrifft. Dadurch nimmt die Streuung der Zwischenankunftszeiten wiederum zu.

Bei Verwendung von Hardware mit einer anderen CPU-Leistung bzw. bei Verwendung von Prozess-Priorisierung verändert sich die Streuung der Jitter ebenfalls. Das Auftreten der Sender-Jitter und deren Abhängigkeit von der Paketrate bleibt aber grundsätzlich erhalten. Bei Messungen auf Systemen mit einem P III 1 GHz-Prozessor waren beispielsweise die Standardabweichungen der Sender-Jitter um eine Größenordnung höher

als bei den Messungen mit einem P IV 3 GHz-Prozessor. Dabei war die Häufigkeit des Auftretens von großen Zwischenankunftszeiten in etwa gleich, die Werte, um die die Zwischenankunftszeiten gestiegen sind, aber vielfach höher (siehe auch Kapitel 5.1.2.1).

5.1.1.2 Poisson-Quelle

Für die Evaluierung der Poisson-Quelle wird die Verteilung der Zwischenankunftszeiten der Datenpakete beim Sender untersucht. Erwartet wird eine Exponentialverteilung mit der Verteilungsfunktion gemäß Gleichung (5.1), wobei λ die Paketrate der Datenquelle ist [Jai91, Kap.30.4]:

$$p(t) = \lambda \cdot e^{-\lambda t} \tag{5.1}$$

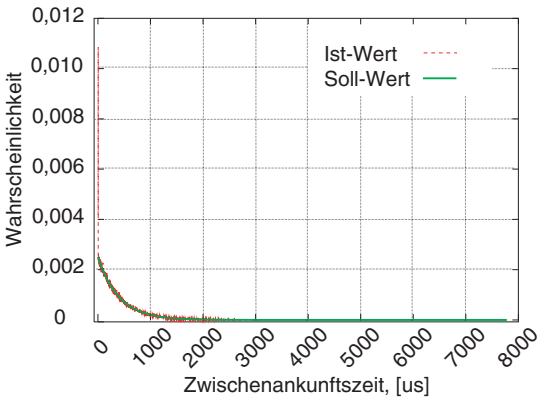

Abbildung 5.11: Wahrscheinlichkeitsverteilung der Zwischenankunftszeiten der Datenpakete bei einer Poisson-Quelle

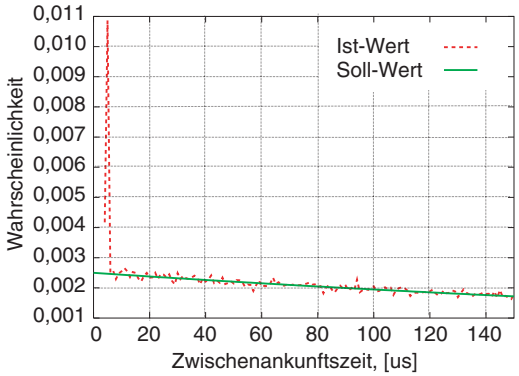

Abbildung 5.12: Ein Ausschnitt aus der Wahrscheinlichkeitsverteilung der Zwischenankunftszeiten bei einer Poisson-Quelle

In den Abbildungen 5.11 und 5.12 wird die Wahrscheinlichkeitsverteilung der Zwischenankunftszeiten einer Messung mit der Datenrate 20 MBit/s bei einer Paketlänge von 1.000 Byte, was einer Paketrate von $\lambda = 2.500$ Pakete/s entspricht, dargestellt. Dabei stellt die Abbildung 5.12 einen Ausschnitt der Verteilungsfunktion im Bereich geringer Zwischenankunftszeiten dar. Eine Messdauer von 1.000 s ist vorgegeben worden. Den Abbildungen kann entnommen werden, dass die gemessenen Zwischenankunftszeiten die zugrunde liegende Exponentialverteilung relativ gut nachbilden. Eine Abweichung der Wahrscheinlichkeit vom vorgegebenen Quellenmodell um ca. 0,01 ergibt sich lediglich im Wertebereich 0–9 µs. Diese ist im minimalen zeitlichen Abstand zwischen zwei benachbarten Datenpaketen auf dem Link begründet.

Aus den Eigenschaften der Poisson-Quelle wird bei der oben genannten Paketrate ein Mittelwert sowie eine Standardabweichung der Zwischenankunftszeiten gleich dem

Kehrwert der Paketrate – 400 µs – erwartet. Die gemessenen Werte sind der Tabelle 5.3 zu entnehmen. Offensichtlich wird der Mittelwert auf die Mikrosekunde genau eingehal-

Größe	**Wert**
Dauer der Session [s]	1.000,1
Anzahl gesendeter Datenpakete	2.501.514
mittlere Datenrate des Senders [MBit/s]	20,01
Anzahl verlorener Datenpakete	0
Paketverlustrate [%]	0
mittlere Zwischenankunftszeit im Sender [µs]	400
minimale Zwischenankunftszeit im Sender [µs]	4
Maximale Zwischenankunftszeit im Sender [µs]	81.321
Standardabweichung der Zwischenankunftszeiten am Sender [µs]	404

Tabelle 5.3: Zusammenfassung der Messergebnisse mit der Poisson-Datenquelle

ten. Die Standardabweichung der Zwischenankunftszeiten ist 1 % höher als erwartet, was durch die Abweichung der Zwischenankunftszeiten bei ganz geringen Zwischenankunftszeiten verursacht sein kann. Des Weiteren wurde für die quantitative Bewertung für das Einhalten des vorgegebenen Quellenmodells der mittlere quadratische Fehler der gemessenen PDF gegenüber den theoretisch berechneten Werten ermittelt. Es ergibt sich ein Wert von $1,53 \cdot 10^{-4}$.

5.1.1.3 FIFO-Quelle

Für den Test der Funktion der FIFO-Quelle wird in einem Perl-Skript eine externe Quelle mit einer Gleichverteilung der Zwischenankunftszeiten der Datenpakete im Bereich 500 – 2.500 µs realisiert. Die Paketlänge wird konstant auf 1.000 Byte gesetzt.

Die Ergebnisse der Berechnung der Aussendezeiten werden in eine *named pipe* geschrieben, aus welcher der Reader-Thread von *LTest* die Sendemarken entnimmt. Zu den vorgegebenen Zeitpunkten versucht der Sender-Thread ein Datenpaket vorgegebener Größe zu senden. Die Ergebnisse der Untersuchung der Zwischenankunftszeiten sind in Tabelle 5.4 sowie in Abbildung 5.13 dargestellt.

Erwartungsgemäß ergibt die CDF im Bereich 500 – 2.500 µs annähernd eine Gerade zwischen den Werten 0 und 1 auf der Ordinate. Ähnlich wie bei der CBR-Quelle liegen 0,03 % der Datenpakete außerhalb des vorgegebenen Bereichs. Dies ist ebenfalls durch

Größe	Wert
Dauer der Session [s]	150,63
Anzahl gesendeter Datenpakete	100.000
mittlere Datenrate des Senders [MBit/s]	5,31
Anzahl verlorener Datenpakete	2.417
Paketverlustrate [%]	2,417
mittlere Zwischenankunftszeit im Sender [µs]	1506
minimale Zwischenankunftszeit im Sender [µs]	5
maximale Zwischenankunftszeit im Sender [µs]	4.703
Standardabweichung der Zwischenankunftszeiten am Sender [µs]	579

Tabelle 5.4: Zusammenfassung der Messergebnisse der FIFO-Quelle mit gleichverteilten Zwischenankunftszeiten

den Busy-Wait-Scheduler verursacht. Der mittlere quadratische Fehler der gemessenen CDF gegenüber den theoretisch berechneten Werten beträgt $1,04 \cdot 10^{-3}$.

5.1.2 Prozess-Priorisierung

Für die Untersuchung der Auswirkung der Prozess-Priorisierung auf die Güte des Systems werden in einem Punkt-zu-Punkt-Szenario folgende Messreihen durchgeführt:

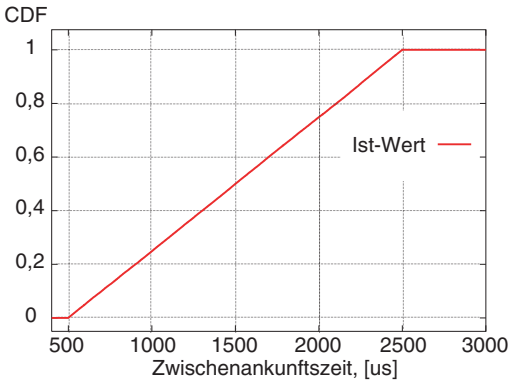

Abbildung 5.13: CDF der Zwischenankunftszeiten mit einer externen FIFO-Quelle mit einer Gleichverteilung von 500 – 2.500 µs

1. Messung ohne Prozess-Priorisierung, wobei die Hintergrund-Last auf dem sendenden System schrittweise von 0 auf 75 % erhöht wird.

2. Messung mit einer Priorisierung von *LTest*, wobei die Hintergrund-Last auf dem sendenden System schrittweise von 0 auf 75 % erhöht wird.

3. Messung mit einer Prozess-Priorisierung, wobei die Hintergrund-Last auf dem empfangenen System schrittweise von 0 auf 75 % erhöht wird.

Um die Unterschiede zwischen dem Verhalten eines belasteten und eines unbelasteten Systems deutlich hervorzuheben, werden bei der Untersuchung der Auswirkung der Priorisierung im Sender weniger performante Rechner verwendet als bei sonstigen Untersuchungen. Dafür werden im Messaufbau gemäß Abbildung 5.1 die Rechner *Dellmess1* und *Dellmess2* durch zwei Rechner mit einer Pentium III CPU mit einer Taktfrequenz von 1 GHz ersetzt. Hierdurch sind höhere Werte der Sender-Jitter als in den vorangehenden Untersuchungen zu erwarten. Im Vordergrund der Betrachtung steht das unterschiedliche Verhalten des Systems mit und ohne Prozess-Priorisierung.

Die Datenübertragung erfolgt über UDP. Es wird mit einer CBR-Quelle mit Paketen der Länge 150 Byte gesendet. Für das Erzeugen einer definierten Prozessor-Last wird das Programm *Loadgen* verwendet (siehe Anhang B).

5.1.2.1 Prozess-Priorisierung im Datensender

In den Messreihen 1 und 2 wird die Auswirkung der Priorisierung im Sender untersucht. Abbildungen 5.14 und 5.15 stellen die Ergebnisse dar, wobei auf unterschiedliche Skalierung der y-Achse in den Abbildungen zu achten ist. In der Abbildung links ist das

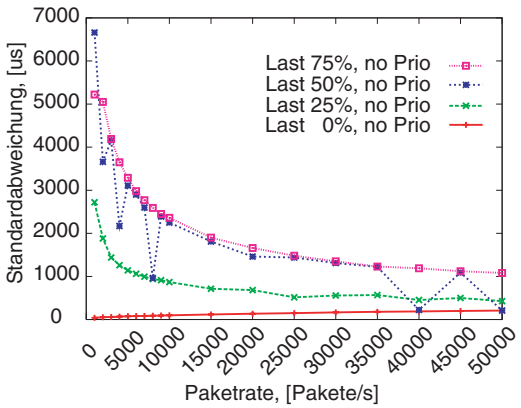

Abbildung 5.14: Zwischenankunftszeiten am Sender ohne Prozess-Priorisierung

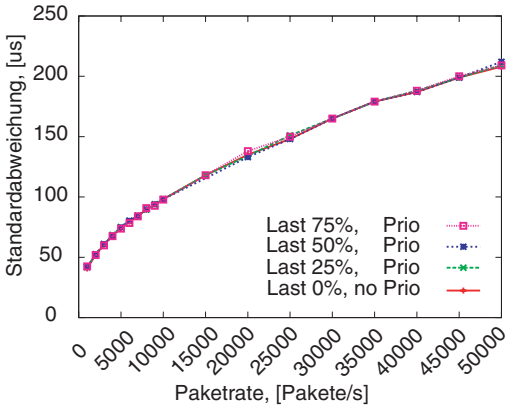

Abbildung 5.15: Zwischenankunftszeiten am Sender mit Prozess-Priorisierung

Sinken der Güte der Datenquelle mit steigender Hintergrund-Last deutlich erkennbar. Vor allem bei Messungen mit einer Hintergrund-Last sinkt die Güte der Datenquelle bei kleinen Paketraten besonders stark (siehe Kapitel 5.1.1). Wird der Sender-Thread priorisiert, so ist die Güte der Datenquelle unabhängig von der Hintergrund-Last (Abbildung rechts). Damit bestätigt sich die Wirksamkeit der Prozess-Priorisierung im *LTest*-Sender.

Um die Abhängigkeit von der Hardware-Performance darzustellen, wurde dieselbe Messung mit den Rechnern *Dellmess1* und *Dellmess2* durchgeführt (Pentium IV CPU, 3 GHz). Die Ergebnisse sind in Abbildung 5.16 dargestellt. In Abbildung 5.17 ist der Vergleich zwischen der Messung auf dem System mit einer Pentium III CPU und der mit einer Pentium IV CPU ohne Hintergrund-Last und ohne Priorisierung dargestellt. Hier ist deutlich die Verringerung der Standardabweichung der Sender-Jitter erkennbar.

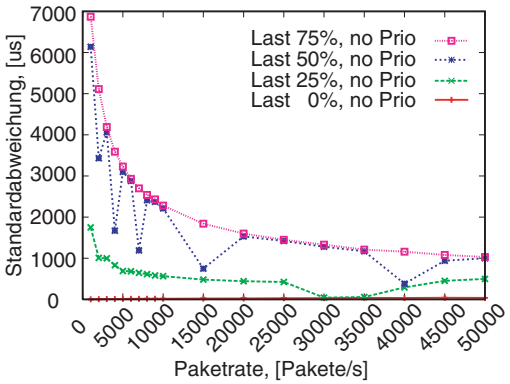

Abbildung 5.16: Zwischenankunftszeiten am Sender ohne Prozess-Priorisierung, P IV

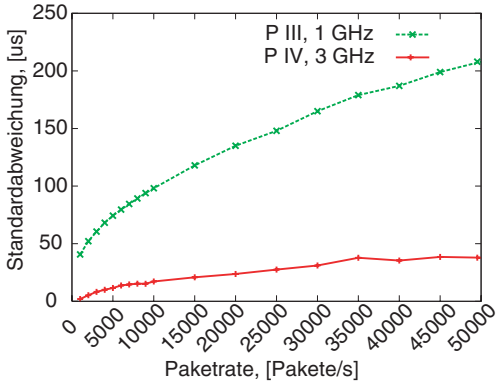

Abbildung 5.17: Güte des Senders in Abhängigkeit von verwendeter Hardware

Bei der Messung mit Hintergrund-Last von 25 % sind etwas geringere Streuungen der Sender-Jitter als bei dem schwächeren Rechner feststellbar. Bei höheren Lasten ist hingegen auf höher performanter Hardware keine bessere Güte der Datenquelle feststellbar. Damit wird bestätigt, dass die Güte der Datenquelle in erster Linie durch Abschaltung von Hintergrund-Prozessen verbessert wird. Die Hardware-Performance hat vor allem bei einem unbelasteten System Auswirkung auf die Güte der Datenquelle.

Des Weiteren ist in Abbildungen 5.18 und 5.19 ein Vergleich der Messungen mit und ohne Priorisierung von *LTest* bei einer Paketrate von 1.000 Pakete/s und einer Hintergrund-Last von 75 % dargestellt Diese basiert auf den Messungen mit den Rechnern mit der Pentium III CPU. Die gewählte Paketrate entspricht dem Messpunkt mit der jeweils maximalen Standardabweichung der Zwischenankunftszeiten der ersten Messreihe. Die Auswirkung der Prozess-Priorisierung ist hier daran erkennbar, dass die Häufigkeit für das Auftreten sehr hoher bzw. sehr geringer Zwischenankunftszeiten bei einer Messung mit Prozess-Priorisierung vielfach geringer ist als bei der Messung oh-

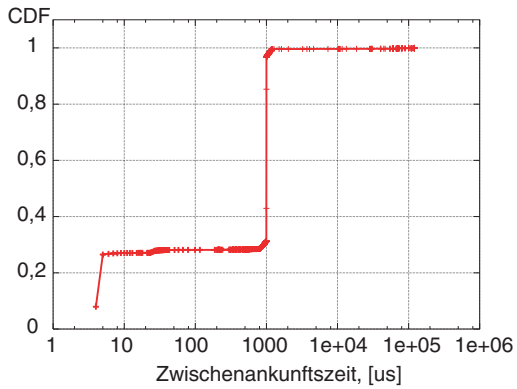

Abbildung 5.18: CDF der Zwischenankunftszeiten am Sender bei 75 % Hintergrund-Last ohne Priorisierung von *LTest*

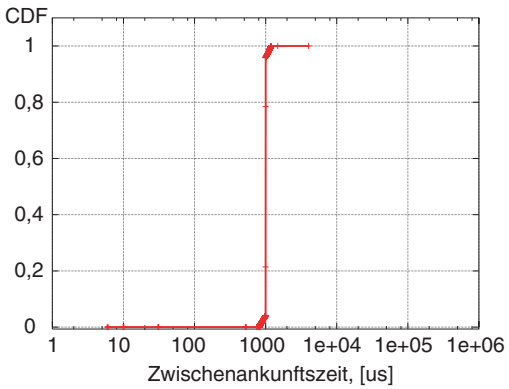

Abbildung 5.19: CDF der Zwischenankunftszeiten am Sender bei 75 % Hintergrund-Last mit Priorisierung von *LTest*

ne Verwendung der Prozess-Priorisierung. Wie in Kapitel 5.1.1.1 angenommen, werden Jitter bei Verwendung einer Prozess-Priorisierung im Sender stark reduziert, aber nicht ganz eliminiert. So liegt das Maximum der Zwischenankunftszeit ohne Verwendung der Prozess-Priorisierung in der beschriebenen Messung bei 140 ms, bei Verwendung von Prozess-Priorisierung sinkt das Maximum auf 4,7 ms.

5.1.2.2 Prozess-Priorisierung im Datenempfänger

Bei der Untersuchung der Auswirkung der Prozess-Priorisierung im Empfänger kann die Standardabweichung der Zwischenankunftszeiten der Datenpakete am Empfänger betrachtet werden. Die Paketrate wird auf der Abszisse eingetragen. Die Ordinate zeigt die für die jeweilige Messung berechnete Standardabweichung (siehe Abbildung 5.20).

Zu beachten ist, dass hier die Standardabweichung der Zwischenankunftszeiten im Empfänger in Relation zu der im Sender zu betrachten ist (die untere Linie in der Abbildung). Der Einfluss der Hintergrund-Last auf die Zwischenankunftszeiten ist wesentlich geringer als im Sender, aber trotzdem deutlich erkennbar. Dies ist in einer vielfach geringeren CPU-Last begründet, welche der *LTest*-Empfänger im Gegensatz zum Sender erzeugt. Außerdem ist zu beachten, dass die Streuung der Zwischenankunftszeiten mit der Priorisierung des Empfänger-Threads auch bei Hintergrund-Last geringer ist als im unbelasteten System ohne Prozess-Priorisierung.

Abbildung 5.21 zeigt die Datenrate am Empfänger in Abhängigkeit von der Paketrate am Sender für unterschiedliche Werte der Hintergrund-Last. Aus der Abbildung ist erkennbar, dass bei der Messung mit Hintergrund-Last und ohne Prozess-Priorisierung ein leichtes Absacken der Datenrate im Empfänger mit steigender Sender-Datenrate festzustellen ist. Das bedeutet, dass bei steigender Paketrate ohne Prozess-Priorisierung

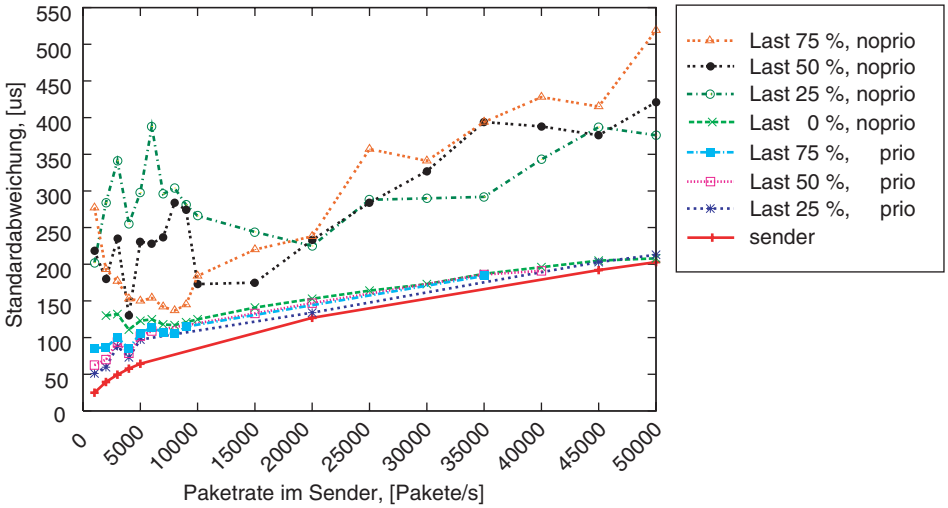

Abbildung 5.20: Zwischenankunftszeiten beim Empfänger in Abhängigkeit von der Hintergrund-Last mit und ohne Prozess-Priorisierung

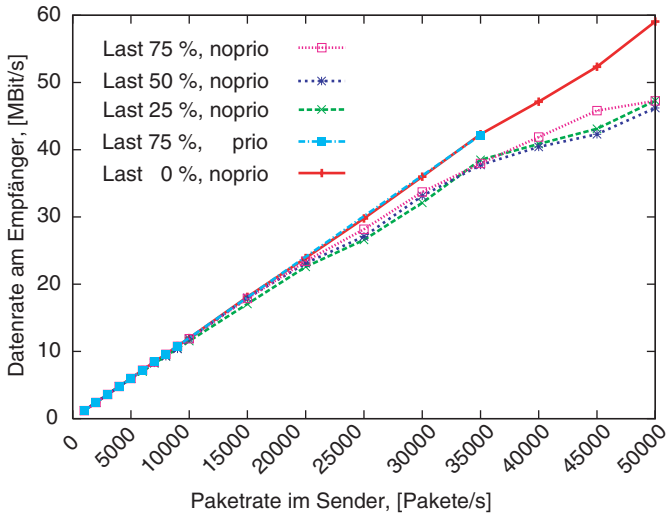

Abbildung 5.21: Datenrate beim Empfänger in Abhängigkeit von der Hintergrund-Last sowie von der Prozess-Priorisierung

der Paketverlust schneller ansteigt als bei Verwendung einer Priorisierung. Dies deutet darauf hin, dass der Empfänger eine längere Zeit benötigt, um die Datenpakete vom Socket zu lesen als bei einem priorisierten Thread. Folglich ist anzunehmen, dass die Verzögerung zwischen dem tatsächlichen Empfang eines Datenpakets und dem Setzen des Empfänger-Zeitstempels ansteigt. Daraus ist anzunehmen, dass der Fehler τ_{sched} (siehe Kapitel 2.5.2) durch die Verwendung einer Prozess-Priorisierung reduziert wird. Die Verringerung des Fehlers τ_{sched} ist außerdem in der Abbildung 5.22 an der Reduktion der Höhe der in regelmäßigen Abständen auftretenden Spitzen der Verzögerungen (ca. 13 ms in der grünen Kurve, ca. 2 ms bei der roten) deutlich erkennbar. Die Ursachen für das Auftreten dieser Spitzen werden in Kapitel 5.1.4 bei der Abschätzung des Fehlers τ_{sched} diskutiert.

5.1.2.3 Seiteneffekte

In den vorhergehenden Kapiteln wurde gezeigt, dass durch die Prozess-Priorisierung die Güte der Datenquelle verbessert sowie der Fehler τ_{sched} reduziert werden kann. Die Verwendung einer Prozess-Priorisierung weist aber einige Seiteneffekte auf.

Um den *LTest*-Prozess priorisieren zu können, muss das Programm mit Root-Rechten laufen. Dadurch wird das System stärker fehleranfällig als beim Ablauf mit Rechten eines nicht privilegierten Nutzers. Wird z.B. beim Messen mit den Rechten des Superusers der gesamte Arbeitsspeicher aufgebraucht (z.B. durch die Speicherung der *LTest*-Header), so kommt das Rechnersystem zum Absturz.

Außerdem kann die Systemuhr beim Messen mit Root-Rechten einen sehr hohen Schlupf aufweisen. Dies Verhalten wird in der Abbildung 5.22 dargestellt. Hier wird der zeitliche Verlauf der gemessenen Verzögerung der Datenpakete in der oben diskutierten Messung (1.000 Pakete/s, 75 % Hintergrund-Last) dargestellt.

Beide beteiligten Rechner werden auf ein DCF77-Signal synchronisiert. In der Abbildung sind die gemessenen Verzögerungen über die Dauer einer Messung von 20 s dargestellt. Bei der Messung mit Prozess-Priorisierung (rote Kurve) ist deutlich die Verringerung der Einweg-Verzögerungen über die Messdauer erkennbar.

Zwischen beiden Rechnern befinden sich keine aktiven Komponenten, die am Anfang der Messung Datenpakete gespeichert hätten. Folglich kann diese kontinuierliche Reduktion der Verzögerung nur in einer Differenz des Schlupfes beider Uhren – dem relativen Schlupf – verursacht sein. Aus der Differenz der mittleren Verzögerungen am Anfang und am Ende der Messung kann diese Differenz berechnet werden. Diese Berechnung wird nachfolgend vorgestellt:

Die Differenz der mittleren Laufzeiten der Messung im Anfangsbereich ($t_1 = 0\ s,\ D_1 =$

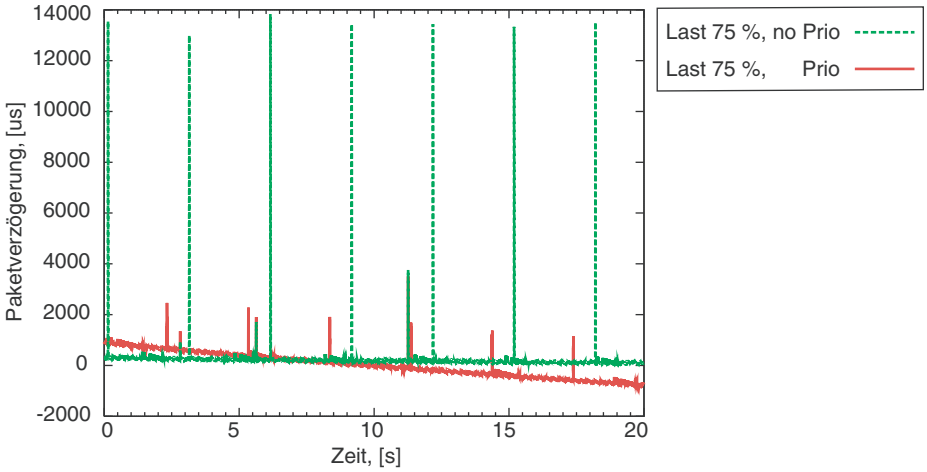

Abbildung 5.22: Veränderung der Verzögerung im Laufe einer Messung mit und ohne Priorisierung des Empfänger-Threads

900 μs) und Endbereich (($t_2 = 20\ s,\ D_2 = -730\ \mu s$) beträgt in 20 s:

$$D_2 - D_1 = 900 - (-730) = 1630\ \mu s$$

Daraus ergibt sich ein relativer Schlupf der Uhren von:

$$\frac{1630 \cdot 10^{-6}\ s}{20\ s} = 8,15 \cdot 10^{-5}$$

Bei der Messung ohne Prozess-Priorisierung ist ebenso ein Schlupf der Uhr feststellbar. Die mittlere Verzögerung am Anfang der Messung ist ca. um 200 µs höher als am Ende des Messvorgangs. Damit lässt sich ein relativer Schlupf der Uhren von

$$\frac{200 \cdot 10^{-6}\ s}{20\ s} = 1 \cdot 10^{-5}$$

berechnen. Dies entspricht der in Kapitel 3.2.4.1 gegebenen Abschätzung des Schlupfes der Uhren. Aus der Richtung der Neigung der Kurve kann festgestellt werden, dass die Uhr am Empfänger-System der am Sender nachläuft.

5.1.3 Schlussfolgerungen

In den Kapiteln 5.1.1 und 5.1.2 sind die Sender-Jitter sowie Jitter im Empfänger bzw. deren Verteilung betrachtet worden. Speziell die Standardabweichung der Zwischenankunftszeiten zeigt sich als gut geeignet für eine quantitative Bewertung der Einhaltung eines vorgegebenen Quellenmodells.

Die vorgestellten Ergebnisse zeigen die Güte der Einhaltung der vorgegebenen Aussendezeiten der Datenpakete von *LTest*. Die Ergebnisse in Kapitel 5.1.1.1 zeigen, dass die Wahrscheinlichkeit für eine Abweichung um mehr als 10 µs von den vorgegebenen Aussendezeiten den Wert $1,5 \cdot 10^3$ nicht überschreitet (siehe Tabelle 5.2). Die Maxima der Abweichungen können allerdings je nach Paketrate und Auslastung des Systems mehr als 10 ms betragen. Diese punktuell auftretenden hohen Sender-Jitter sind durch die Unterbrechung des Sender-Threads von *LTest* verursacht.

Bei der Planung von Netzmessungen ist außerdem zu berücksichtigen, dass bei Datenraten von mehr als 80 – 90 % der Link-Datenrate eine genaue Nachbildung des gewünschten Quellenmodells nicht möglich ist.

Unter Berücksichtigung der Hardware-Architektur, bei welcher die Interrupt-Verzögerungen im Bereich von 2–3 µs liegen und kein deterministisches Prozess-Scheduling (keine harte Echtzeitfähigkeit) zum Einsatz kommt, kann dies als ein sehr gutes Ergebnis gewertet werden. Es ist aber zu beachten, dass keine deterministischen Garantien gegeben werden können, dass der Messfehler eine bestimmte Grenze nicht überschreitet.

Die Ergebnisse können durch bestimmte Maßnahmen verbessert werden – z.B. durch eine Prozess-Priorisierung oder durch einen weitgehenden Verzicht auf sonstige Hintergrund-Prozesse im System. Störungen, verursacht durch konkurrierende Anwendungsprozesse sowie Kernel-Threads, können aber nicht gänzlich eliminiert werden. Zu beachten ist, dass mit einem echtzeitfähigen Betriebssystem auf den eingesetzten PC- bzw. SPARC-Rechnerarchitekturen Sender-Jitter im Bereich von mehreren Mikrosekunden, z.B. durch Interrupt-Verzögerungen und durch den Ablauf von Kernel-Threads, zu erwarten sind [Pas03, Kap.7]. Mit dem gewählten Ansatz zur Implementierung von *LTest* wird somit eine Güte der Quelle erreicht, die nahe an die durch die gewählte Rechnerarchitektur vorgegebenen Grenzen herankommt.

5.1.4 Einweg-Verzögerungen und Abschätzung des Fehlers τ_{sched}

Bei der Bewertung eines Messsystems ist die Abschätzung der Fehler der Messung der Einweg-Verzögerungen sowie der abgeleiteten Größen von besonderer Bedeutung. Der Fehler τ_{offset} (siehe Kapitel 2.5.2) ist von der Güte der Uhrensynchronisation des Messsystems abhängig. In Kapitel 3.2.5.1 werden Verfahren zur Uhrensynchronisation diskutiert. Des Weiteren sind die Anforderungen an die Genauigkeit der Verzögerungsmessung wesentlich geringer als bei der Messung von Jittern. Mit einer in Kapitel 3.2.4.1 genannten Stabilität der Uhren sowie einer Zwischenankunftszeit $\delta_t \ll 1\ s$ von Datenpaketen ergibt sich ein Fehler $\tau_{offset}(t)$ bei der Messung von Jittern in Höhe von $\tau_{offset}(t) \ll 10$ µs. Es wird erwartet, dass der Grenzwert für τ_{sched} mindestens im Bereich von 100 µs liegt und somit ausschlaggebend bei der Fehlerbetrachtung ist.

Für die Abschätzung von τ_{sched} wird eine weitere Messreihe mit dem Aufbau gemäß Abbildung 5.1 durchgeführt. Gemessen wird dabei über UDP mit einer CBR-Quelle und einer Paketlänge von 1.500 Byte. Jede Messung dauert 20 s. Die Datenrate wird schrittweise von 120 kBit/s auf 1 GBit/s erhöht. Nachfolgend werden die gemessenen Einweg-Verzögerungen betrachtet. In den Abbildungen 5.23 bis 5.27 wird eine Auswahl aus den Messergebnissen dargestellt.

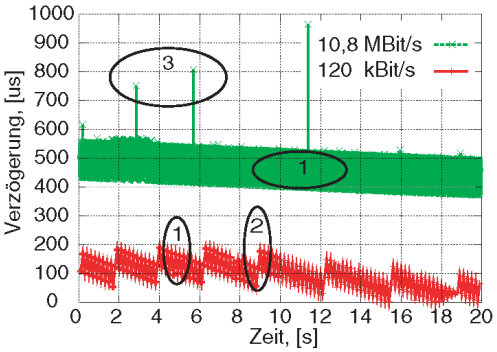

Abbildung 5.23: Verteilung gemessener Einweg-Verzögerungen, Datenraten 120 kBit/s und 10,8 MBit/s

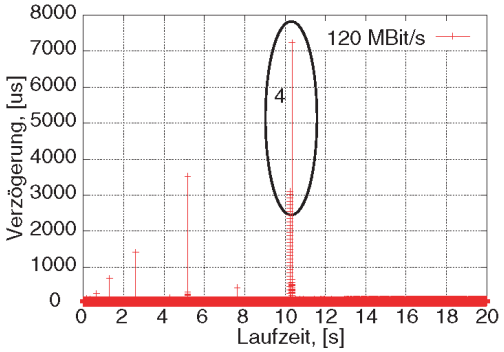

Abbildung 5.24: Verteilung gemessener Einweg-Verzögerungen, Datenrate 120 MBit/s

In den ersten beiden Abbildungen werden die gemessenen Verzögerungen über die Messdauer bei unterschiedlichen Sender-Raten gezeigt. Die Auswirkung der Uhrendrift ist in der Veränderung der Verzögerung über die Zeit deutlich erkennbar. Es handelt sich beim Messfehler, verursacht durch den relativen Schlupf der Uhren, offensichtlich um einen systematischen Fehler. Dieser kann bei der Auswertung der Ergebnisse eliminiert werden (siehe Kapitel 3.2.4.1).

Um die markanten Schwankungen der gemessenen Verzögerungen zu referenzieren, sind einige Bereiche in den Abbildungen 5.23 und 5.24 mit den Ziffern 1 bis 4 markiert. Bis auf einige wenige Ausnahmen, welche nachfolgend gesondert betrachtet werden, liegen die Kurzzeit-Schwankungen der Laufzeiten im Bereich 80 bis 120 µs (siehe Abbildung 5.23). Diese Bereiche sind mit der Ziffer 1 markiert. Diese Schwankungen treten ebenfalls in Abbildung 5.24 auf, sind aber wegen einer groben Skalierung der Darstellung nicht direkt erkennbar.

Des Weiteren finden in zeitlichen Abständen von $2 - 4$ Sekunden weitere Sprünge der Einweg-Verzögerung um ca. $120 - 150$ µs statt. Diese sind in den Ergebnissen der Messung mit 120 kBit/s (rote Kurve in Abbildung 5.23) besonders deutlich erkennbar. Einer dieser Sprünge ist mit der Ziffer 2 markiert.

Die oben genannten Schwankungen der Laufzeiten können nicht durch das Suspendieren des *LTest*-Prozesses bei dem Sender oder dem Empfänger verursacht worden sein,

weil die Dauer der Zeitscheiben für einzelne Prozesse in der Größenordnung von Millisekunden liegt. Außerdem ist bei einer entsprechend feinen zeitlichen Auflösung der Darstellung das Muster der Laufzeit-Schwankungen erkennbar – es haben immer jeweils zwei benachbarte Datenpakete eine Differenz der gemessenen Laufzeiten, die durch die lokalen Maxima und Minima der Kurven repräsentiert werden. Mit steigender Daten- bzw. Paketrate nimmt die Amplitude dieser Schwankungen in geringem Maße zu (grüne Kurve in Abbildung 5.23 sowie rote in Abbildung 5.24). Die Periode der hochfrequenten Schwankungen nimmt in etwa umgekehrt proportional zur Paketrate ab. Das beschriebene Verhalten des Messsystems deutet auf eine Bufferung der Datenpakete in der Netzwerk-Hardware der Endsysteme oder im IP-Stack hin. Es ist nahe liegend anzunehmen, dass die hochfrequenten Schwankungen durch das Buffern in der Hardware verursacht worden sind und die Sprünge alle 2-4 Sekunden (markiert mit der Ziffer 2) durch das Buffern der Datenpakete im Betriebssystem verursacht sind.

Für die Betrachtungen an dieser Stelle ist es unerheblich, ob die Ursache der Schwankungen in der Netzwerk-Hardware oder im Betriebssystem liegt, weil in beiden Fällen die Schwankungen von der Anwendung nicht beeinflusst werden können. Vielmehr wird die beschriebene Beobachtung in die Bewertung der Messfehler in Kapitel 5.1.4.2 einbezogen. Von besonderem Interesse sind die wenigen Fälle, in denen die gemessene Verzögerung $D \gg 100$ µs liegt (am Beispiel der Marken 3 und 4). In den durchgeführten Messungen treten solche Verzögerungen im Mittel nicht häufiger als alle 2 – 5 Sekunden auf. Tendenziell steigen die Werte solcher Ausreißer mit steigender Übertragungsrate bis auf ca. 7 – 8 ms. In Abbildung 5.25 ist das Ergebnis einer Messung mit annähernd maximal erreichbarer Datenrate auf dem Link dargestellt. Die Werte der Maxima der Verzögerungen steigen offensichtlich zwischen der Messung mit 120 MBit/s und 820 MBit/s nicht mehr. Die Häufigkeit des Auftretens solcher Spitzen der Laufzeiten liegt in derselben Größenordnung wie bei Messungen mit einer geringeren Datenrate. Die Häufigkeit des Auftretens solcher Spitzen ändert sich bei einer Hintergrund-Last im System nicht wesentlich (vgl. mit Abbildung 5.22).

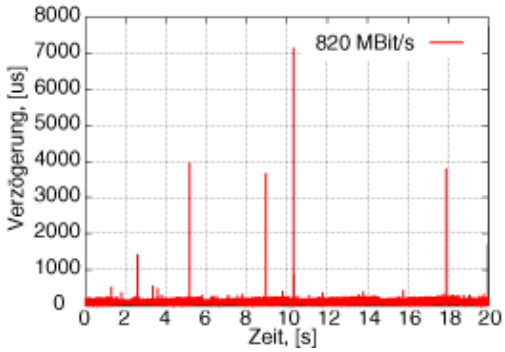

Abbildung 5.25: Verteilung gemessener Einweg-Verzögerungen, Datenrate 820 MBit/s

Werden die gemessenen Laufzeiten der Datenpakete in unmittelbarer Umgebung dieser Spitzen mit einer hohen zeitlichen Auflösung dargestellt, so wird deutlich erkennbar, dass die Darstellung eines der zwei charakteristischen Muster aufweist. In den Abbildungen 5.26 und 5.27 sind die zwei Muster mit einer entsprechend hohen zeitlichen

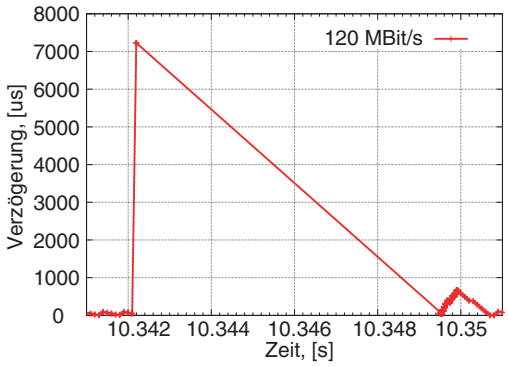

Abbildung 5.26: Auswirkung der Task-Unterbrechung im Sender

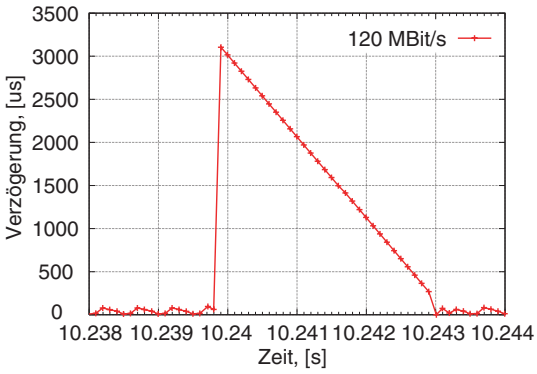

Abbildung 5.27: Die Auswirkung der Task-Unterbrechung im Empfänger

Auflösung dargestellt, die bei der Messung mit der Datenrate 120 MBit/s aufgetreten sind (siehe Marke 4 in Abbildung 5.24).

Auf der Abszisse sind die Aussendezeiten der Datenpakete eingetragen. In der Abbildung links ist der Wert der Einweg-Verzögerung in der Spitze in etwa gleich dem zeitlichen Abstand zum nachfolgenden Datenpaket. Auf die hohe Verzögerung des Datenpakets folgt eine lange Pause bis zum Aussenden des nachfolgenden Datenpakets. Daraus kann man schlussfolgern, dass diese hohe Verzögerung direkt mit dem Sender-Jitter in Verbindung steht. Es sind offensichtlich keine Buffer gefüllt, da das direkt der Spitze nachfolgende Datenpaket dieselbe Laufzeit hat wie die Datenpakete vor dieser Spitze. Diese gemessene Verzögerung kann somit nur durch eine Task-Unterbrechung zwischen dem Setzen des Zeitstempels und dem Aussenden eines Datenpakets im *LTest*-Sender verursacht sein. Das Paket wird erst ausgesendet, nachdem die CPU dem *LTest*-Prozess wieder zugewiesen wird. Aus diesem Grund ist die Verzögerung in der Spitze in etwa gleich dem Abstand zum benachbarten Datenpaket.

Um die mittlere Datenrate einzuhalten, werden die nachfolgenden Datenpakete vom Datensender mit voller Link-Rate gesendet. Diese Übertragung mit voller Datenrate verursacht eine kurzzeitige Stauung der Datenpakete im Buffer des Empfängers. Diese Stauung ist in der zweiten (kleineren) Spitze direkt im Anschluss an die erste erkennbar. Ebenso ist bei einer höheren zeitlichen Auflösung der Darstellung erkennbar, dass die Datenpakete während des Anstiegs in der zweiten Spitze mit voller Link-Rate ankommen.

Im Gegensatz zum oben dargestellten Muster werden in Abbildung 5.27 die Datenpa-

kete durchgehend äquidistant gesendet. Die Datenpakete nach der Spitze weisen eine stetige Abnahme der gemessenen Laufzeit auf. Die Differenz der Einweg-Verzögerungen benachbarter Datenpakete entspricht in etwa der Zwischenankunftszeit der Datenpakete am Sender (100 µs bei der dargestellten Messung). Es wird an dieser Stelle angenommen, dass dieses Muster der Einweg-Verzögerungen durch die Unterbrechung des Empfänger-Threads verursacht ist. Ist diese Annahme richtig, so ist folgendes Verhalten der Verzögerungen in der Spitze zu erwarten:

Für die Dauer der Unterbrechung des *LTest*-Empfängers werden keine Datenpakete an die Anwendung weitergereicht. Der *LTest*-Sender sendet nach wie vor die Datenpakete zeitlich äquidistant aus. So lange der Buffer im IP-Stack des Betriebssystems nicht überläuft, werden die Datenpakete im IP-Stack des Empfängers zwischengespeichert. Bekommt *LTest* wieder die CPU zugewiesen, so werden die in der Zeit der Inaktivität empfangenen Datenpakete unmittelbar nacheinander an *LTest* weitergereicht und mit einem Empfänger-Zeitstempel versehen. Der Buffer wird somit entleert, wobei die Differenz der gemessenen Laufzeit benachbarter Datenpakete der Zwischenaussendezeit im Sender abzüglich der Verarbeitungszeit der PDU im *LTest*-Empfänger entspricht. Genau dieses Verhalten stellt die Abbildung 5.27 dar.

Somit kann festgestellt werden, dass die punktuell auftretenden Einweg-Verzögerungen von $D \gg 100$ µs Messfehler darstellen, welche durch das Suspendieren des Sender- bzw. Empfänger-Threads verursacht sind. Erwartungsgemäß treten Messfehler, welche dem *LTest*-Empfänger zuzuordnen sind, vielfach seltener auf als die durch den *LTest*-Sender verursachten Fehler. Dies liegt an der vollen CPU-Auslastung durch den Busy-Wait-Scheduler im *LTest*-Sender.

5.1.4.1 Unterschiede in Ergebnissen bei UDP- und TCP-Messungen

Der wesentliche Unterschied bei den Messungen über UDP und TCP ergibt sich primär aus der Verwendung einer Fehlerkontrolle und Fehlerkorrektur im TCP. Damit sind wesentliche Unterschiede bei Messergebnissen in erster Linie in Szenarien mit hohen Verzögerungen und Verlusten zu erwarten. In den bislang betrachteten Messszenarien ist bewusst der Einfluss des Netzes minimiert worden, um die Interaktion der Anwendung mit dem Betriebssystem und der Hardware hervorzuheben. Somit sind bezüglich der Güte der Datenquelle im oben verwendeten Szenario keine signifikanten Unterschiede bei der Verwendung von TCP und UDP zu erwarten. Dies wird auch in den früher vorgestellten Abbildungen 5.9 und 5.10 bestätigt.

Bei der Betrachtung der Einweg-Verzögerungen wird dagegen erwartet, dass bei TCP-basierten Messungen eine breitere Streuung der Einweg-Verzögerungen auftritt. Der Unterschied der Einweg-Verzögerungen bei Verwendung von UDP und TCP ist in den Abbildungen 5.28 und 5.29 deutlich erkennbar.

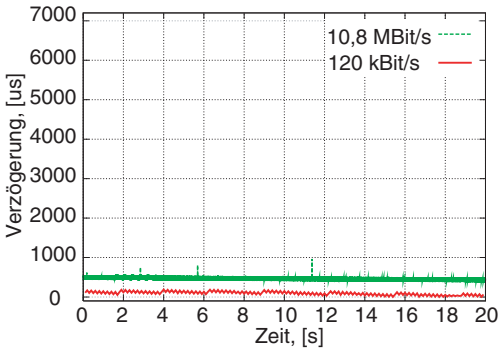

Abbildung 5.28: Gemessene Einweg-Verzögerungen bei Verwendung von UDP

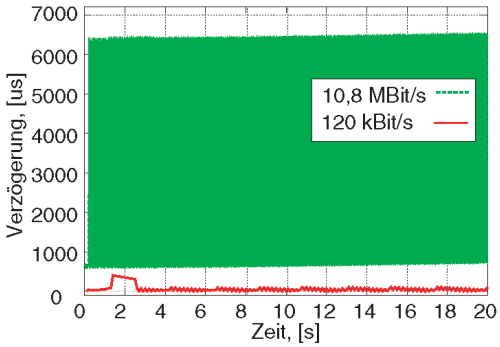

Abbildung 5.29: Gemessene Einweg-Verzögerungen bei Verwendung von TCP

Gesichtspunkte, unter welchen die Wahl des Transportprotokolls für die Messungen stattfinden soll, werden in Kapitel 5.4 diskutiert.

5.1.4.2 Schlussfolgerungen

Die in Kapitel 5.1.4 vorgestellten Ergebnisse zeigen, dass mit *LTest* eine Untersuchung der Jitter im Bereich von 100 µs möglich ist. Es liegt aber ein beträchtlicher Unterschied in den Mittelwerten der Einweg-Verzögerungen bei den Messungen mit unterschiedlichen Datenraten vor (siehe Abbildungen 5.23 und 5.24). Dieser liegt im Bereich von 100 bis 500 µs. Die einzelnen Messungen waren dabei im Abstand von $10-15$ Minuten voneinander durchgeführt worden. Daraus folgt, dass eine Langzeit-Synchronisation der Uhren auf 50 µs (unter Verwendung der PZF77-Uhren) nicht erreicht wird.

Wird eine genauere Bestimmung des relativen Offsets der Uhren zu einem bestimmten Zeitpunkt benötigt, so ist der Offset z.B. mit den in Kapitel 3.2.5.3 sowie in [Pax98b] vorgestellten Methoden zu ermitteln. Anschließend kann der ermittelte Offset als systematischer Fehler von den gemessenen Ergebnissen subtrahiert werden. Die Stabilität der Uhren bei den verwendeten Systemen liegt in den in Kapitel 3.2.4.1 angenommenen Grenzen von $1 \cdot 10^{-5}$.

Abweichend von dem hier vorgestellten Ergebnissen zeigen Untersuchungen in [Grö04] und [Bie04], dass auf vereinzelten Rechnern die Systemuhren sehr starke Schwankungen der Drift aufweisen. Diese Instabilität war bei den davon betroffenen Systemen immanent vorhanden. Eine Abhilfe hat in beiden Arbeiten nur ein Austausch der Hardware gebracht. Aus diesen Beobachtungen folgt die Notwendigkeit, die Stabilität der Systemuhren der verwendeten Rechner im Vorfeld zu überprüfen.

Ebenso wie bei der Betrachtung der Sender-Jitter kann bei der Abschätzung der Messfehler der Einweg-Verzögerungen und der Einweg-Jitter nur eine statistische

Abschätzung des Fehlers gegeben werden. Für die Abschätzung des Fehlers τ_{sched} ist noch festzulegen, welche Komponenten der oben betrachteten Verzögerungen dem Fehler zuzurechnen sind.

In der Definition der Einweg-Verzögerung nach RFC 2679 (siehe Kapitel 2.3.1 sowie [Alm99a]) wird lediglich die Zeit zwischen dem Aussenden des ersten Bits auf die Leitung bis zum Empfang des letzten Bits von der Leitung betrachtet. Die vorgestellten Schwankungen der Verzögerung, verursacht in den Buffern der Netzwerk-Interfaces und des IP-Stacks, sind damit dem Fehler τ_{sched} zuzurechnen. Diesem sind ebenso die gemessenen Verzögerungen, welche die Unterbrechung des Sender- bzw. Empfänger-Threads von *LTest* darstellen, zuzurechnen.

Spitzen der Laufzeiten von mehr als 800 µs treten in den vorgestellten Messungen mit einer Häufigkeit von $2 \cdot 10^{-5} - 6 \cdot 10^{-5}$ auf. Bei Messungen auf Systemen mit einer geringeren Taktfrequenz der CPU sowie bei Messungen mit Hintergrund-Last liegt die Häufigkeit der Spitzen in derselben Größenordnung. Dafür verändert sich deren Höhe in Abhängigkeit von der Performance sowie der Auslastung des Systems wesentlich. Damit liegt der Fehler τ_{sched} mit einer Wahrscheinlichkeit ca. 99,9994 % im Bereich von 80 – 150 µs. Mit einer Restwahrscheinlichkeit von bis zu $6 \cdot 10^{-3}$ % liegt der Fehler im Bereich einiger weniger Millisekunden. Das Maximum des Fehlers hängt dabei von der CPU-Performance, von der Höhe der Hintergrund-Last sowie von der Übertragungsrate ab. Die genauen Werte des Fehlers können je nach verwendeter Hardware und Messszenario empirisch, z.B. mit den oben beschriebenen Messungen, bestimmt werden.

Die Wahrscheinlichkeit für das Auftreten des Fehlers im Bereich von einigen wenigen Millisekunden stellt die Grenze für die Realisierung eines hochpräzisen Messsystems auf einem Betriebssystem ohne harte Echtzeitfähigkeit dar. Die hier dargestellte Abschätzung des Fehlers betrifft die Messung der Verzögerung im Netz ohne Einbeziehung der Endsysteme. Folglich ist diese Betrachtung der Netzwerk-Performance zuzuordnen.

Es ist aber zu beachten, dass es gerade bei der Betrachtung der Ende-zu-Ende-Dienstgüte häufig erwünscht ist, dass die Verweildauer in den Buffern der Endsysteme in der Einweg-Verzögerung mit berücksichtigt wird. Hierfür ist die Definition der Dienstgüte nach ITU-T, wie in Kapitel 2.1.1 beschrieben, besser geeignet, weil diese sich auf die Dienstgüte in einer bestimmten Schicht des OSI-Stacks bezieht. Bei einer anwendungsbezogenen Betrachtung der Dienstgüte in IP-Netzen ist es hier zweckmäßig, die Verzögerung der Datenpakete oberhalb der Transport-Schicht des TCP/IP-Stacks zu betrachten. In diesem Fall sind lediglich die oben vorgestellten Spitzenwerte der Verzögerung in den obigen Messungen als Fehler τ_{sched} zu berücksichtigen. Somit tritt ein Messfehler im Bereich von wenigen Millisekunden mit einer Wahrscheinlichkeit $2 - 6 \cdot 10^{-3}$ % auf, sonst liegt der Fehler in der Größenordnung von 1 μs. Das Maximum dieser Fehler hängt ebenfalls von der Hardware-Performance, vom verwendeten

Betriebssystem sowie von der Auslastung des Systems und von der Datenrate des Mess-Datenstroms ab.

5.2 Darstellung der Notwendigkeit für das Blockieren des LTest-Servers

Der Blocking-Mode wurde eingeführt, um den Nutzern des Systems die Möglichkeit zu geben, den *LTest*-Server für eine einzelne Messung zu blockieren (siehe Kapitel 4.2.1). Um die Notwendigkeit des Blocking-Mode zu verdeutlichen, wird nachfolgend der gegenseitige Einfluss mehrerer Mess-Datenströme in einer Messung mit konkurrierenden Datenströmen zu einem *LTest*-Server gezeigt. Es wird dabei in einem Szenario gemäß Abbildung 5.30 gemessen.

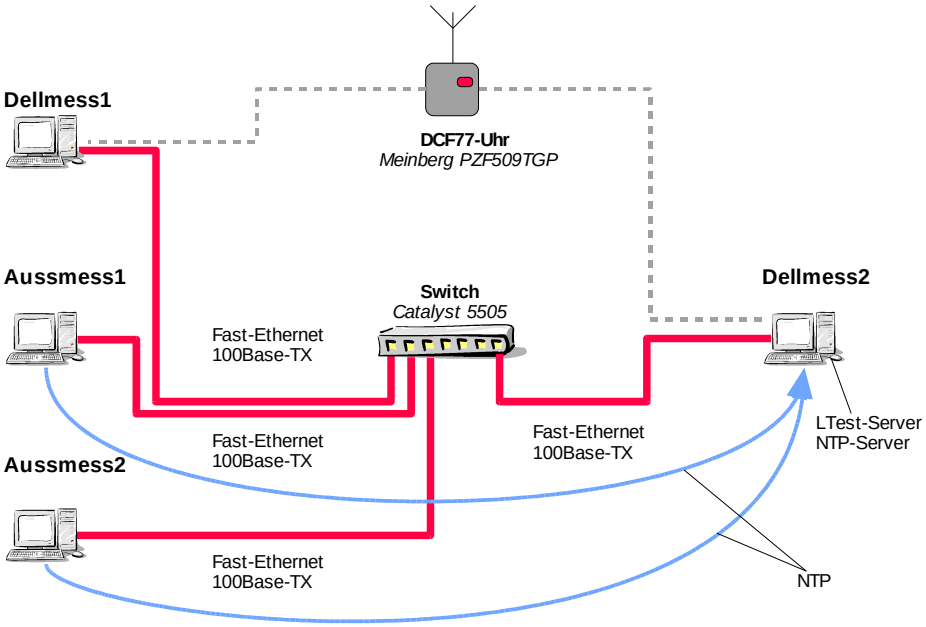

Abbildung 5.30: Messaufbau für das Messen mit konkurrierenden Datenströmen

Zusätzlich zu den bislang verwendeten Rechnern *Dell Precision 360* (*Dellmess1* und *Dellmess2*) kommen zwei PCs mit einer Pentium III CPU und einer Taktfrequenz 1 GHz (*Aussmess1* und *Aussmess2*) zum Einsatz. Diese dienen der Erzeugung konkurrierender Datenströme. Alle vier PCs werden über einen Switch *Catalyst 5505* über Fast-Ethernet-Ports, Full-Duplex, miteinander verbunden. Die Rechner *Dellmess1* und *Dellmess2* werden über DCF77 synchronisiert. Die anderen PCs werden über NTP auf *Dellmess2* synchronisiert.

Als Erstes wird eine einzelne Messreihe ohne konkurrierende Datenströme durchgeführt. Das verwendete Protokoll ist UDP, die Paketlänge ist 150 Byte. Gemessen wird zwischen *Dellmess1* und *Dellmess2*. Die Datenrate der Messung wird von 240 kBit/s schrittweise auf 42 MBit/s erhöht. Dies entspricht einer Paketrate zwischen 200 und 35.000 Pakete/s. Jede einzelne Messung dauert 20 s.

In einer weiteren Messreihe wird zu demselben Empfänger gleichzeitig von *Dellmess1* sowie von *Aussmess1* und *Aussmess2* mit denselben Datenraten wie in der ersten Reihe gemessen. Da der Link auf der Empfängerseite ebenfalls eine Fast-Ethernet-Verbindung mit 100 MBit/s ist, wird dieser beim Senden mit jeweils 33,3 MBit/s einschließlich aller Protokoll-Header sowie Ethernet-Trailer vollständig gefüllt. Bei einer *LTest*-PDU-Größe von 150 Byte entsteht durch die UDP-, IP- und Ethernet-Header sowie den Ethernet-Trailern ein Overhead von 36 %. Dies entspricht einer in *LTest* vorgegebenen Datenrate von 24,5 MBit/s. Bei der zweiten Messreihe wird nur das Ergebnis zwischen *Dellmess1* und *Dellmess2* ausgewertet.

In Abbildung 5.31 wird der Paketverlust in Abhängigkeit von der Sender-Rate für beide Messreihen dargestellt. Mit einer roten senkrechten Linie ist die Sender-Rate, bei welcher der Link bereits zu 100 % gefüllt wird, gekennzeichnet.

Bei einem einzelnen Datenstrom treten in der ganzen Messreihe keine nennenswerten Paketverluste auf – der maximale Paketverlust beträgt beim Senden mit 42 MBit/s etwa 0,1 %. Bei konkurrierenden Strömen treten erste Paketverluste bei einem Empfänger bereits bei einer Senderate von etwa 19 MBit/s auf, was einer Auslastung des Links von etwa 73 % entspricht. Ein wesentlicher Paketverlust tritt erst ab einer Datenrate 24 MBit/s auf, was einer Auslastung des Links von ca. 98 % entspricht (siehe Abbildung 5.31). Hinsichtlich der Paketverluste werden also die Messergebnisse durch die konkurrierenden Messungen nur minimal beeinflusst. Bei der Betrachtung der Streuung der Einweg-Verzögerungen können bereits ab einer Sender-Rate von 6 MBit/s deutliche Unterschiede zwischen den beiden Messreihen festgestellt werden (siehe Abbildung 5.32). Dies ist im Anstieg der Standardabweichung der Einweg-Verzögerungen erkennbar. Solche gegenseitigen Störungen der Mess-Datenströme können durch das Blockieren des *LTest*-Servers bei der Messung vermieden werden. Damit wird die Güte der Messergebnisse erhöht.

5.3 Performance des Systems

Nachfolgend werden diverse in der Laborumgebung erreichten Parameter vorgestellt, welche die Performance des Systems beschreiben.

Auf einer direkten Gigabit-Ethernet-Verbindung zwischen zwei PCs gemäß Abbildung 5.1 wird eine maximale Übertragungsrate von 960 MBit/s sowohl mit TCP als

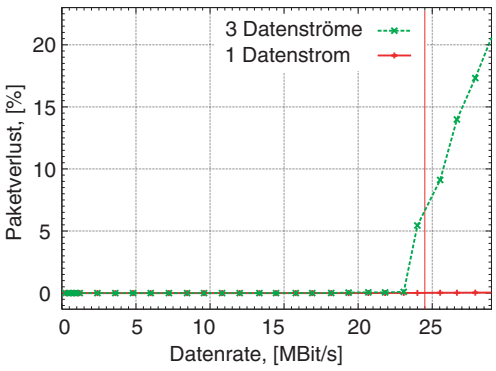

Abbildung 5.31: Paketverluste in Abhängigkeit von der Sender-Rate bei der Messung mit einzelnen und konkurrierenden Datenströmen

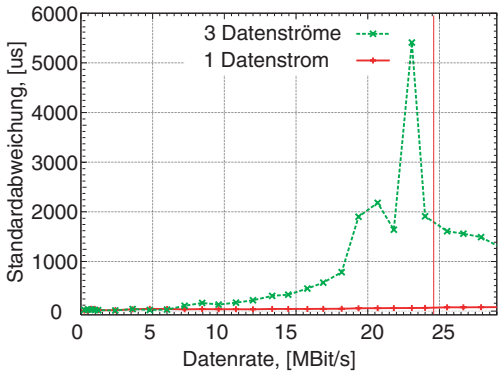

Abbildung 5.32: Standardabweichung der Einweg-Verzögerung in Abhängigkeit von der Sender-Rate bei der Messung mit einzelnen und konkurrierenden Datenströmen

auch mit UDP auf der Empfängerseite erreicht. Dabei liegen bei UDP-Messungen die Paketverluste deutlich unter 1 %. Unter Berücksichtigung der Overheads der Protokollrahmen der Schichten 2 bis 4 entspricht dies einer Auslastung des Links von ca. 98 %. Mit einem zwischengeschalteten PC-basierten Switch treten bei dieser Datenrate unter Verwendung des UDP-Protokolls Paketverluste von $4,5$ % auf, welche in der steigenden Burstiness bei hohen Datenraten ihre Ursache haben (siehe [Bie04]).

Bei Messungen mit einer großen Anzahl an Probepaketen ist die limitierende Komponente die Kapazität des Arbeitsspeichers im System. Während der Untersuchungen sind Messungen mit bis zu 30 Mio. Probepaketen in einer Session durchgeführt worden. Dabei belegt der *LTest*-Client 764 MByte Arbeitsspeicher. Bei der genannten Anzahl an Probepaketen belegen die *LTest*-Header ohne Overhead 720 MByte. Der Overhead bei der Speichernutzung liegt also bei < 6 %.

Bei der Festlegung der Sender-Rate der CBR-Quelle entsteht bei Verwendung von hohen Paketraten ein relativ großer Rundungsfehler. Dieser wird durch die interne Umrechnung der Datenrate in eine Zwischenankunftszeit verursacht, wobei der Abstand auf volle Mikrosekunden gerundet wird. So sind z.B. in der Messreihe aus Kapitel 5.1.4 im Bereich der vollen Link-Datenrate keine Sender-Raten zwischen 923 MBit/s und 1.000 MBit/s möglich, da diese beiden Datenraten bei der verwendeten Paketlänge von 1.500 Byte der Zwischenankunftszeit von 14 bzw. 13 µs entsprechen. Dieser Fehler könnte durch die Verwendung einer zeitlichen Auflösung in Nanosekunden verringert werden. Dies würde aber einen wesentlich höheren Rechenaufwand in der Datenquelle bedeuten, was deren Güte wiederum verringern würde. Ebenso könnte man zum Einhalten der vorgegebenen Datenrate in regelmäßigen Abständen zusätzliche Pakete aussenden, dies aber würde zu einer systematischen Erhöhung der Sender-Jitter führen.

Neben den bereits vorgestellten Ergebnissen wurden Tests auf einem System *Sun Enterprize 450* mit zwei Prozessoren und 4 GByte RAM sowie auf einem *Suse-Linux*-System (Version 8.2) mit 2 Prozessoren Xeon 2,8 GHz und 4 GByte RAM durchgeführt. Es wurde kein stark abweichendes Verhalten des Messystems auf dieser Hardware festgestellt. Da beide Systeme nicht dediziert für Tests zur Verfügung standen, wurden auf diesen keine besonders performante Untersuchungen durchgeführt.

5.4 Wahl geeigneter Messszenarien für aktive Einweg-Messungen

Bevor zwei typische Messszenarien diskutiert werden, sollen nachfolgend Ergebnisse einer Untersuchung in einem produktiven Netz vorgestellt werden. Der Messaufbau dafür ist in Abbildung 5.33 skizziert. Die beiden Messsysteme wurden jeweils vor und hinter einem Router *Cisco 7500* über einen Gigabit-Ethernet-Link angeschlossen. Dieser Router dient als Gateway zwischen einem Intranet mit mehreren tausend Rechnern und dem Übergang ins Internet. Wegen der vielen eingetragenen Access-Listen auf dem

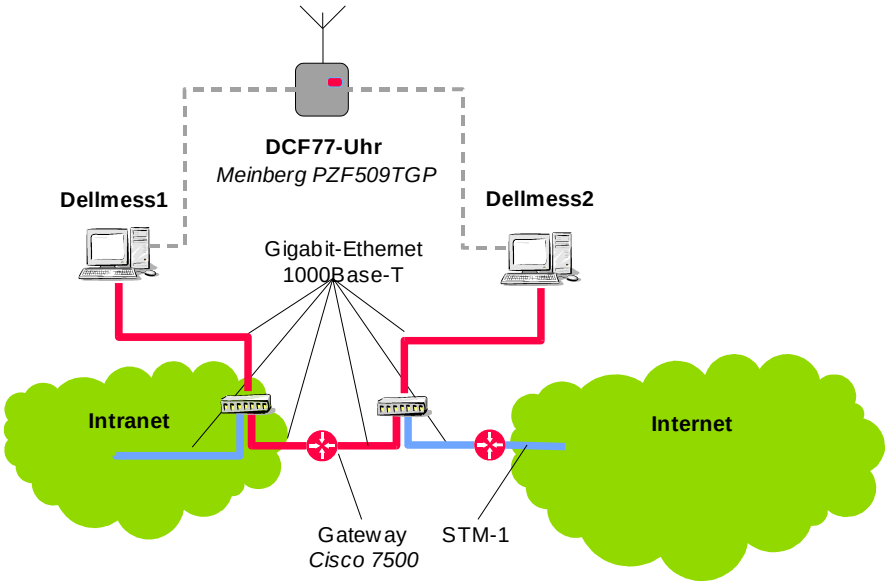

Abbildung 5.33: Das Messszenario für die Performance-Untersuchung des Gateways zum ISP

Router hat dieser zeitweise eine hohe Prozessor-Last und stellt gewissermaßen einen Flaschenhals der Verbindung zwischen dem Intranet und dem Internet dar.

Über eine Zeitdauer von 43 Stunden – von ca. 21:00 Uhr am Montag, den 05.07.2004, bis 16:00 Uhr am Mittwoch, den 07.07.2004 – wurden parallel zwei Messreihen mit einer Sender-Datenrate von 100 kBit/s durchgeführt. Bei der einen wurden Datenpakete der

Länge 150 Byte, bei der anderen Datenpakete der Länge 1.400 Byte übertragen. Jede einzelne Messung dauerte 57 Minuten und wurde zu jeder vollen Stunde gestartet. Die dreiminutigen Pausen wurden genutzt, um die Ergebnisse auf die Festplatte zu speichern.

Zur gleichen Zeit wurden alle zehn Minuten Kurzzeit-Messungen mit einer Paketlänge von 1.450 Byte und einer Datenrate von 300 MBit/s durchgeführt. Diese Messungen dauerten jeweils 20 Sekunden. Alle Messungen wurden über UDP durchgeführt. Ziel dieser Untersuchung war, die Abhängigkeit der Ergebnisse vom gewählten Messszenario aufzuzeigen sowie die Möglichkeit zur Bestimmung der Netzwerk-Performance in diesen Messszenarien vorzustellen. Zu beachten war, dass die Engpässe bei der CPU-Performance und nicht bei der verfügbaren Link-Datenrate zu erwarten waren. Die Ergebnisse der Messungen sind in den Abbildungen 5.34 bis 5.36 dargestellt.

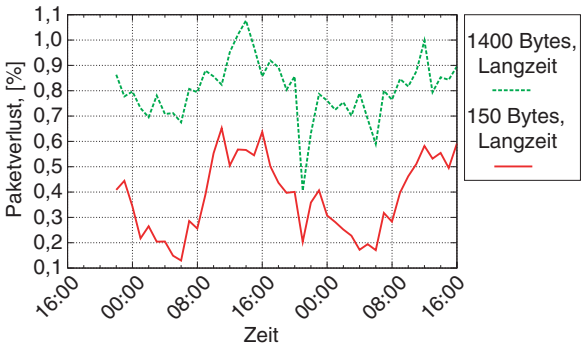

Abbildung 5.34: Paketverluste im Langzeit-Messszenario

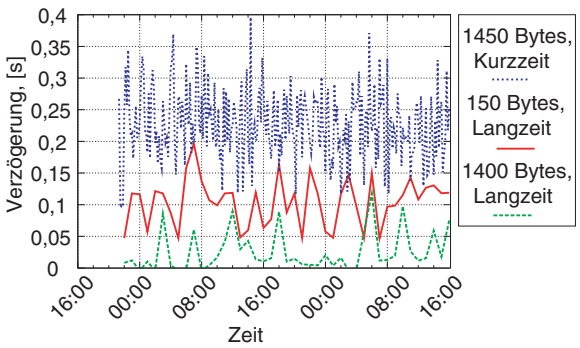

Abbildung 5.35: Maxima der Einweg-Verzögerungen im Beobachtungszeitraum

In den Abbildungen 5.34 und 5.35 ist deutlich die unterschiedliche Behandlung großer und kleiner Datenpakete im Router zu sehen. Große Datenpakete werden häufiger verworfen als kleine. Kleine Datenpakete haben dagegen eine höhere maximale Verweildauer in der Input-Queue des Routers. Aus der Darstellung ist z.B. erkennbar, dass vor allem tagsüber die Netzwerk-Performance für Videokonferenz-Anwendungen (große Datenpakete) zu gering sein kann. Für Sprachanwendungen wie z.B. IP-Telefonie wäre die Netzwerk-Performance, vor allem wegen geringerer Verlustraten von kleinen Datenpaketen, ausreichend.

Aus der Abbildung 5.35 ist außerdem der Unterschied der Maxima der Paketverzögerungen zwischen Langzeit-Messungen mit einer geringen Datenrate und Kurzzeit-Messungen mit hoher Datenrate ersichtlich. Bei den Kurzzeit-Messungen treten außerdem starke Paketverluste – je nach Tageszeit zwischen 53 % und 65 % – auf (siehe Abbildung 5.36). Dabei ist die limitierende Komponente nicht die Datenrate der Interfaces bzw. der Gigabit-Ethernet-Verbindungen, sondern die CPU-Performance der

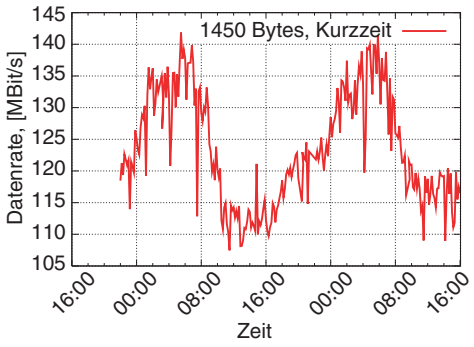

Abbildung 5.36: Datenrate am Empfänger bei Kurzzeit-Messungen

Netzwerk-Prozessoren des Routers. Aus der oben dargestellten Tatsache, dass kleine Datenpakete seltener verworfen werden als große, dafür aber längere Verweildauern in den Queues erfahren (siehe Abbildungen 5.34 und 5.35), folgt, dass bei den durchgeführten Messungen die Paketverluste wegen Warteschlangenüberlauf verworfen werden. Damit wäre zu erwarten, dass die Maxima der Einweg-Verzögerungen durch die Länge der Input-Queue limitiert sind und somit bei allen Messungen gleich sind. Dies entspricht aber nicht den dargestellten Messergebnissen. Folglich ist das Maximum der Verarbeitungszeit mindestens so groß wie die maximale Differenz der Maxima unterschiedlicher Kurzzeit-Messungen und liegt im Bereich von mehr als 100 ms (vgl. Kapitel 2.4.3, Seite 30).

In Abbildung 5.34 ist die Abhängigkeit der Paketverluste von der Tageszeit relativ schwach ausgeprägt. Dies kann daran liegen, dass jeder Messpunkt die statistische Analyse einer Messung von rund einer Stunde darstellt und damit die Dynamik der Schwankungen der Auslastung des Systems zu stark geglättet wird. Im Unterschied dazu ist bei den in Abbildung 5.36 dargestellten Ergebnissen der Reihe der Kurzzeit-Messungen der Tagesverlauf der Datenrate am Empfänger wesentlich deutlicher erkennbar. Ebenso ist erkennbar, dass sich die Maxima der Verzögerungen innerhalb von jeweils 10 Minuten stark verändern.

Aus den vorgestellten Messungen mit einem Messaufbau gemäß Abbildung 5.33 ist die unterschiedliche Behandlung von kleinen und großen Datenpaketen in Netzknoten deutlich erkennbar. Ebenso ist die Abhängigkeit der Messergebnisse von dem Messszenario dargestellt. Mit den vorgestellten Messungen wird z.B. belegt, dass der Netzwerk-Prozessor des untersuchten Routers zu schwach dimensioniert ist und somit die limitierende Komponente des Systems darstellt.

Nachfolgend werden einige für die Planung aktiver Einweg-Messungen relevante Aspekte genannt:

- Steht im Vordergrund der Betrachtung die Netzwerk-Performance, so ist die Messung über UDP zu realisieren, um den Einfluss der Flusssteuerung der Messsysteme auf die Ergebnisse zu vermeiden.
- Wird der Durchsatz in einer TCP-Messung ermittelt, so ist diese ausreichend lange – je nach Verbindung mehrere 100 ms bis mehrere Sekunden – durchzuführen, um den Einfluss des Slow-Starts zu minimieren (siehe [Bie04]).
- Das Verdrängen von TCP-Datenströmen durch UDP-Ströme ist zu berücksichtigen. Speziell bei Messungen in produktiven Netzen ist darauf zu achten, dass Messungen, welche die Verbindungen bzw. die aktiven Komponenten im Netz stark belasten, nicht zu lange andauern, um sonstige Datenströme nicht zu stark zu beeinträchtigen.
- Bei der Realisierung von verteilten n : 1-Messungen über UDP können die Messdatenströme Kontrollverbindungen anderer Mess-Sessions beeinträchtigen.

5.4.1 Kurzzeit-Messungen

Kurzzeit-Messungen sind für die Bestimmung der erwarteten Dienstgüte im Netz gut geeignet. Je nach verwendeten Parametern können diese einige Millisekunden bis einige Sekunden dauern. In einem Kurzzeit-Messszenario kann z.B. die Erreichbarkeit von bestimmten Servern im Netz sowie die Güte ihrer Anbindung an das Netz bestimmt werden. Mit nachgebildeten Profilen einzelner Datenströme von Anwendungen – z.B. Signalisierungsströmen multimedialer Anwendungen sowie von Web-Anwendungen – kann die erwartete Güte eines Dienstes im IP-Netz ermittelt werden. Bei der Nachbildung multimedialer stream-orientierter Anwendungen, wie z.B. Video- und Audio-Streaming, können die Messungen einige Minuten dauern. Diese sind trotzdem den Kurzzeit-Messungen zuzuordnen, weil sie der Bestimmung der momentanen Güte der Anwendung im Netz dienen. Messungen über das TCP-Protokoll sind in erster Linie für die Untersuchung der Dienstgüte von TCP-basierten Anwendungen von Interesse.

In Kurzzeit-Messszenarien können ebenso Momentanwerte der Netzwerk-Performance bestimmt werden. In diesem Fall werden häufig Kurzzeit-Messungen in regelmäßigen Abständen wiederholt. Statistische Auswertungen solcher Kurzzeit-Messungen werden analysiert und beispielsweise grafisch aufbereitet oder zur Netzüberwachung eingesetzt, wobei beim Überschreiten bestimmter statistischer Werte gewisse Alarme oder Trigger ausgelöst werden [Sie01a]. Die oben vorgestellte Untersuchung der Performance des Gateways zum Internet kann als Anwendungsbeispiel von wiederkehrenden Kurzzeit-Messungen zur Analyse bzw. zur Überwachung der Netzwerk-Performance dienen.

5.4.2 Langzeit-Messungen

In Langzeit-Messszenarien erstreckt sich eine Messung über mehrere Minuten bis mehrere Stunden, in einigen Fällen können diese auch mehrere Tage dauern. Solche Messungen sind dem Bereich der Überwachung der Netzwerk-Performance zuzuordnen. Diese dienen der langfristigen Netzüberwachung, der Bestimmung von Engpässen sowie von Störungen im Netz. In Langzeit-Messszenarien können Trends der Veränderung der Netzwerk-Performance über mehrere Monate oder Jahre bestimmt werden. Eine Untersuchung der Netzwerk-Performance auf der Teilnehmer-Anschlussleitung der Internet-Service-Provider T-Online und Colt Telecom mit *LTest* wurde bereits in [Sie04a] veröffentlicht.

5.4.3 Einsatzmöglichkeiten für LTest

In den vorhergehenden Kapiteln sind bereits diverse Messszenarien sowie darin erzielte Messergebnisse vorgestellt worden, welche *LTest* charakterisieren. Nachfolgend werden zusammenfassend die Messgrößen aufgelistet, die mit *LTest* durch aktive Einweg-Messungen bestimmt werden können. Dies sind:

- die Einweg-Paketverzögerung
- der Einweg-Jitter
- der Einweg-Paketverlust
- das Paketverlust-Muster
- die Dauer einer verlustfreien Datenübertragung
- der Durchsatz (BTC)

Außerdem kann die Auslastung der Links sowie das Queueing von zwischengeschalteten Paketvermittlungssystemen untersucht werden sowie die Stabilität bzw. der relative Schlupf der Uhren beteiligter Messsysteme. Die entsprechenden Umlauf-Größen können aus den Einweg-Größen mit der in Kapitel 2.3.2 vorgestellten Methode bestimmt werden.

Damit kann *LTest* für die Bestimmung der Netzwerk-Performance sowie der Dienstgüte sowohl in lokalen als auch in Weitverkehrsnetzen eingesetzt werden. Außerdem kann das Programm in Labor-Umgebungen für die Untersuchung der Performance von Netzknoten bzw. Paketvermittlungssystemen eingesetzt werden.

Kapitel 6

Zusammenfassung und Ausblick

Der wesentliche Beitrag der vorliegenden Arbeit liegt in der Erarbeitung eines neuen Ansatzes für verteilte hochperformante und hochpräzise Messungen der Netzwerk-Performance sowie der Dienstgüte in IP-basierten Netzen. Von vergleichbaren Ansätzen hebt sich dieser durch die Realisierung einer Möglichkeit zum aktiven Messen mit unterschiedlichen Quellenmodellen ab. Neu ist der Ansatz für die Durchführung verteilter Messungen in Hochgeschwindigkeitsnetzen. Durch die Wahl eines geeigneten Kommunikationsprotokolls können mehrere Mess-Clients von unterschiedlichen Orten simultan oder zeitversetzt Messungen zu einem Server durchführen und die Messergebnisse von diesen erhalten. Durch die Einführung des Blocking-Mode kann dabei eine exklusive Nutzung des Servers sichergestellt werden. Der Ansatz für das Messen zwischen zwei entfernten Messsystemen ist bislang in keinen anderen Messsystemen gemacht worden.

Bei der Erstellung des Konzeptes zur Realisierung der Messungen ist eine Analyse der Parameter, welche die Messergebnisse beeinflussen, durchgeführt worden. Bei der Bestimmung der Messgrößen lag der Fokus bei der Betrachtung der Dienstgüte in Verbindung mit der Implementierung von Anwendungen mit Echtzeit-Anforderungen in IP-Netzen. Dazu gehören unter anderem Anwendungen audiovisueller Kommunikation sowie das Grid-Computing. Das in dieser Arbeit vorgestellte Konzept kann aber ebenso für die Überwachung der Netzwerk-Performance sowie für die Bestimmung der Dienstgüte von herkömmlichen Applikationen, wie z.B. Web, Mail und News, angewendet werden.

Das Anwendungsprogramm *LTest* für die Durchführung verteilter aktiver Einweg-Messungen ist im Rahmen der Arbeit entworfen und prototypisch implementiert worden. In dieses sind die erarbeiteten Konzepte eingeflossen. In den in Kapitel 4 vorgestellten Untersuchungen ist die Notwendigkeit getroffener Maßnahmen sowie deren Auswirkung auf die Messergebnisse vorgestellt worden. Durch den modularen Aufbau von *LTest* ist die Entwicklung neuer Quellenmodelle von dem Hauptprogramm entkoppelt worden. Dank der Verwendung von CORBA für die Client-Server-Kommunikation ist eine einfache Erweiterung des Kommunikationsprotokolls zwischen einzelnen Instanzen von *LTest* möglich.

Mit Hilfe von Untersuchungen in einer Labor-Umgebung mit *LTest* sind die Grenzen

der Güte einer Realisierung eines hochpräzisen Messsystems auf Standard-PC-Hardware und Software aufgezeigt worden. Mit *LTest* können Laufzeitschwankungen im Netz im Bereich weniger Mikrosekunden beobachtet werden, wenn auch nur statistische Garantien der Fehlergrenzen gegeben werden können. Damit ist es mit *LTest* möglich, die Funktion einzelner Vermittlungssysteme in IP-Netzen zu analysieren. Ebenso ist eine Untersuchung der Performance von IP-basierten Transportprotokollen mit dem entwickelten Messsystem möglich. Die durchgeführten Untersuchungen haben gezeigt, dass mit der gewählten Software und Hardware Vorgänge im Netz mit einer zeitlichen Auflösung von wenigen Mikrosekunden möglich sind. Einer weiteren Erhöhung der Güte des Systems setzt das verwendete Betriebssystem und die gewählte Hardware-Architektur klare Grenzen. Die hier gewonnenen Erkenntnisse sind nicht auf die Bewertung der Güte von *LTest* beschränkt, sondern haben allgemeine Gültigkeit.

Des Weiteren sind Nebenbedingungen, unter welchen eine hohe Güte der Einhaltung des gewünschten Quellenmodells erreicht werden kann, genannt worden.

Mit der Sicherung der Attribute jedes einzelnen gesendeten Datenpakets für die Weiterverarbeitung der Ergebnisse eröffnet sich die Möglichkeit zur problemorientierten Analyse, Auswertung und Darstellung der Messergebnisse. In Verbindung mit der hohen Effizienz der Speicherung der Messergebnisse stellt dies ebenfalls ein Alleinstellungsmerkmal von *LTest* dar.

LTest wird bereits in unterschiedlichen Forschungsarbeiten sowie bei der Analyse der Netzwerk-Performance in diversen Projekten eingesetzt und hat sich dort als stabil und flexibel erwiesen. Der Quellcode der Implementierung ist unter der BSD-Lizenz [OSI04] auf der Homepage des *LTest*-Projektes zum Download freigegeben worden.

6.1 Ausblick

Mit zunehmender Verbreitung der IP-Netze in Forschungs- und Unternehmensnetzen ist davon auszugehen, dass die Themen Bewertung der Dienstgüte sowie Monitoring der Netzwerk-Performance in Zukunft stark an Bedeutung zunehmen werden. Die gegenwärtig beobachtete Zunahme an Grid-Aktivitäten setzt für das Messen der Dienstgüte neue Akzente. Hierbei ist es zwingend erforderlich, bei der Bewertung der Performance des Netzes Rücksicht auf Mechanismen zur Flusssteuerung in den Endsystemen zu nehmen.

Die Trennung der Rollen der Netz- und Dienstanbieter fördert die Entwicklung vielfältiger, miteinander konkurrierender Dienstangebote im Netz, welche auf zwischen den Anbietern und den Kunden vereinbarten SLA basieren. Hieraus entsteht für die Netz- und Dienstanbieter die Notwendigkeit, für die Nutzer ein Monitoring-System der Güte des

genutzten Netzes bzw. der Dienste bereitzustellen. Es ist ein Trend weg von der Geheimhaltung von Performance-Daten seitens der Anbieter in Richtung der Veröffentlichung und Bereitstellung dieser Daten. Als Beleg dafür kann das seit März 2004 öffentlich verfügbare Monitoring-System des G-WiN [Hol04] dienen.

Es ist davon auszugehen, dass solche, einem eingeschränkten Kundenkreis oder öffentlich zugängliche Systeme zum Netz- bzw. Dienstgüte-Monitoring, an Verbreitung gewinnen werden. Damit entsteht die Anforderung einer serverseitigen Auswertung, Weiterverarbeitung und Visualisierung der Ergebnisse. Aus diesem Grund ist in der Implementierung von *LTest* eine Erweiterung um eine solche Auswertung der Ergebnisse vorgesehen worden.

Auf der Basis der prototypischen Implementierung von *LTest* soll es mit den in Kapitel 5 gewonnenen Erkenntnissen möglich sein, eine Implementierung auf einem echtzeitfähigen Betriebssystem zu realisieren, mit welchem die Messfehler deterministisch im unteren Mikrosekunden-Bereich eingegrenzt werden. Derartige Systeme können für die Untersuchung von Vermittlungsvorgängen in aktiven IP-Netzkomponenten notwendig sein. Zurzeit sind derartige präzise Messungen nur mit dedizierten Hardware-Lösungen realisierbar.

Bei *LTest* handelt es sich um ein verteiltes System, welches von einem entfernten Rechner gesteuert werden kann. Folglich ist beim Einsatz des Systems auf Sicherheitsaspekte hinsichtlich der Nutzung zu achten. Dabei stellt der Kommunikationskanal über CORBA die Schnittstelle zur Steuerung dar, welche eines besonderen Schutzes bedarf. Die verwendete CORBA-Implementierung MICO unterstützt eine Verschlüsselung der Kommunikation mit Hilfe von SSL [SSL96]. Damit ist eine Grundlage für die Erweiterung von *LTest* um den Schutz des Systems vor Angriffen gegeben. Da es sich bei *LTest* um einen Prototyp handelt, sind dort keine Mechanismen zur Authentifizierung und Autorisation implementiert worden.

Die Implementierung des SVG-Moduls *LTestView* hat die Einschränkungen bereits existierender Darstellungsprogramme für SVG aufgezeigt. Die Verwendung des SVG-Moduls kann derzeit nicht empfohlen werden. Es ist aber zu erwarten, dass in nächster Zukunft bessere Implementierungen des SVG-Formats in die gängigen Web-Browser eingebaut werden. Einige Verbesserungen in dieser Hinsicht waren bereits in den letzten Monaten festzustellen. So ist mit dem Web-Browser Konqueror in der Version 3.2 mittlerweile eine Skalierung von SVG-Objekten möglich. An der Verbesserung der SVG-Unterstützung in den Web-Browsern des Mozilla-Projektes [Moz04] wird zurzeit ebenfalls aktiv gearbeitet.

Für den produktiven Einsatz des Messsystems ist außerdem eine Portierung auf Windows-Betriebssysteme sowie die Erweiterung um die Unterstützung des IP-Protokolls Version 6 von Interesse.

Anhang A

Sammlung von Flow-Statistiken mit FlowStats

Für die Untersuchung der Charakteristik der Datenströme in IP-Netzen ist ein Tool namens *FlowStats* entwickelt worden (siehe Kapitel 3.2.3.2). Dabei werden im ersten Schritt die Header von IP-Paketen mit einem so genannten Packet-Sniffer, wie z.B. *tcpdump* [Tcp04], aufgenommen und in einer Datei gespeichert. *FlowStats* kann mehrere Dateien, die im *pcap*-Format [Deg04] vorliegen, einlesen und verarbeiten. Dabei werden folgende Felder des IP- sowie des TCP- bzw. UDP-Headers analysiert:

- Quell-IP-Adresse
- Ziel-IP-Adresse
- IP-Protokoll
- Quell-Port
- Ziel-Port
- IP-Länge

Zusätzlich wird der Zeitpunkt der Beobachtung des Datenpakets zur Analyse herangezogen. Anhand dieser Daten wird versucht, die beobachteten Datenpakete einzelnen Datenströmen zuzuordnen. Diese Zuordnung wird anhand der Definitionen 2.3 und 2.4 (Kapitel 2.3.6) durchgeführt. Die maximale Zeitspanne für die Zuordnung der Datenpakete zu demselben Datenstrom (Interpacket-Timer T) wurde auf 12 Sekunden gesetzt. Diesem Wert liegen empirische Untersuchungen zugrunde, die nachfolgend vorgestellt werden.

A.1 Festlegung des Interpacket-Timers

In Kapitel 2.3.6 ist die Definition für einen TCP- sowie einen UDP-Datenstrom gegeben. Dabei wurde der UDP-Datenstrom unter Verwendung einer maximal zulässigen Zeitspanne zwischen zwei Datenpaketen desselben Datenstroms – des Interpacket-Timers

– gegeben. Der Wert für diesen Timer ergibt sich aus der Semantik einzelner Anwendungen. Wird der Wert des Timers zu hoch gewählt, so steigt die Wahrscheinlichkeit dafür, dass mehrere voneinander unabhängige Datenströme fälschlicherweise zusammengefasst werden. Damit werden große zeitliche Abstände zwischen Paketen gezählt, die eigentlich nicht zueinander gehören. Wird der Timer zu klein gewählt, so werden zueinander gehörende Datenpakete unterschiedlichen Datenströmen zugeordnet. Damit wird wiederum die Statistik der Zwischenankunftszeiten verfälscht.

Um einen geeigneten Wert zumindest für eine Gruppe von Anwendungen zu finden, wurde mit dem Hilfswerkzeug *FlowStats* eine Reihe von Untersuchungen durchgeführt. Bei der Anwendung der Untersuchungen auf RTP-basierte Audio- und Video-Datenströme sinkt sprungartig die Anzahl der beobachteten Datenströme beim Überschreiten der Werte von 5 s und 10 s für den Interpacket-Timer. Ein Ergebnis der Untersuchung von Video-Datenströmen ist in Abbildung A.1 dargestellt. Bei der Beobachtung der

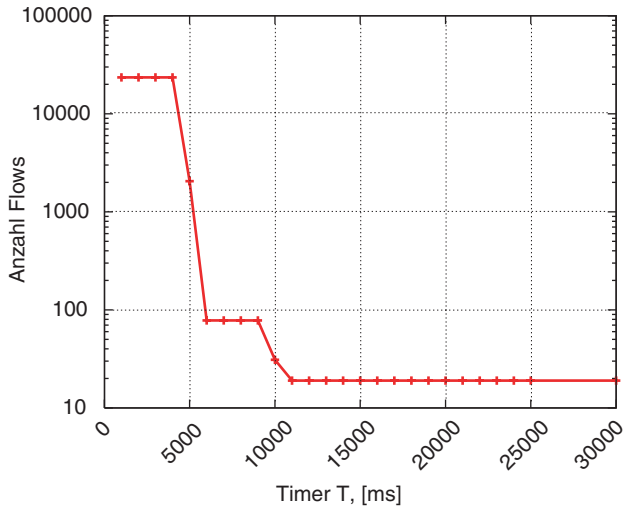

Abbildung A.1: Die Abhängigkeit der Anzahl ermittelter Datenströme vom Wert des Interpacket-Timers

Datenströme an einem VoIP-Gateway, der als Übergang zum ISDN-Netz für die IP-Telefonie-Installation der Universität Hannover dient, wurden sehr ähnliche Ergebnisse erzielt. Am Gateway wurden außerdem Fälle einer Wiederholung der Tupel, die einen Datenstrom identifizieren, in zeitlichen Abständen von 100 bis 200 s beobachtet. Bei sehr stark frequentierten Gateways können die zeitlichen Abstände der Wiederholung der Tupel vielfach geringer sein. Somit darf der Interpacket-Timer nicht zu hoch gesetzt werden.

Im Zusammenhang mit der Bestimmung des Wertes des Interpacket-Timers wurden außerdem punktuell Untersuchungen am G-WiN-Anschluss der Universität Hannover durchgeführt, um ggf. einen allgemein gültigen Wert für den Interpacket-Timer für

UDP-Datenströme zu finden. Diese haben aber divergierende Resultate gezeigt. Damit ist eine zuverlässige Zuordnung der Datenpakete einzelnen UDP-Strömen mit *FlowStats* nur in Verbindung mit RTP-basierten multimedialen Anwendungen möglich.

Anhang B

Generierung von CPU-Last mit Loadgen

Für die Generierung einer definierten CPU-Last (siehe Kapitel 5.1.2) ist ein Perl-Script *Loadgen* geschrieben worden. Dabei soll es einen vorgegebenen Anteil der Prozessor-Zeit unabhängig von der Taktfrequenz der CPU belegen.

Das System führt in kurzen Tasks einige rechenintensive Operationen durch und legt anschließend eine Pause ein, um im Mittel die gewünschte Prozessor-Last zu erzeugen. Dafür wird in einer Schleife eine aktive Phase abwechselnd mit einer passiven Phase durchlaufen:

```
$PauseTime = 2.000.000; # sleep 2 ms
while (1) {
 do_something($UpperBound);
 Time::HiRes::usleep($PauseTime);
 ...
 }
```

Die aktive Phase besteht aus der Funktion *do_something($UpperBound)*. In dieser wird wiederum in einer Schleife eine Potenz-Berechnung von zwei Zufallszahlen durchgeführt. Der übergebene Parameter gibt die Anzahl der Durchläufe dieser Schleife an. Es wird eine lineare Abhängigkeit zwischen der Anzahl der Durchläufe dieser Schleife und der verbrauchten CPU-Zeit angenommen. Somit kann die in einem Schleifendurchlauf verbrauchte CPU-Zeit über den Parameter *$UpperBound* gesteuert werden. In der passiven Phase wird der Prozess für 2 ms suspendiert. In Realität ist auch die passive Phase wegen der Unsicherheit des Task-Wechsels (siehe Anhang C) variabel.

Nach dem Durchlauf der oben vorgestellten Phasen wird ein Korrekturfaktor für die Variable `$UpperBound` berechnet. Diese wird wie folgt berechnet:

```
$Usage = ($CPUTime / $TimeElapsed) * 100;

# now, correct the number of cicles for doing something
$CorrectFct = 1 - ($Usage - $Percent)/$Percent;
```

```
# upper/lower bound for factor
if ($CorrectFct > 1.6)
 {
    $CorrectFct = 1.6;
 }
    elsif ($CorrectFct < 0.4)
 {
    $CorrectFct = 0.4;
 }
 $UpperBound *= $CorrectFct;
```

Dabei ist `$Percent` die Soll-Last in Prozent, `$CPUTime` die vom Prozess verbrauchte CPU-Zeit, `$TimeElapsed` die reale Zeit seit dem Start des Programms.

Die Eingrenzung des Korrekturfaktors ist notwendig, um starke Oszillationen in der Einschwingphase bzw. infolge von wechselnden Belastungen des Systems durch andere Prozesse zu verhindern.

Um die Funktion von *Loadgen* zu überprüfen, wurde die erzeugte CPU-Last in zwei Szenarien untersucht. In beiden Fällen wird auf einem Rechner mit einer Pentium III - CPU mit der Taktfrequenz 1 GHz gemessen. Im ersten Szenario wird über eine Zeitspanne von 1.000 s eine Last von jeweils 25, 50 und 75 % erzeugt. Das System ist sonst unbelastet. Einmal pro Sekunde wird die vom Programm verbrauchte Prozessor-Zeit im Verhältnis zur verstrichenen Zeit in Prozent sowie der Korrekturfaktor `$CorrectFct` ausgegeben. Die Ergebnisse der Ausgaben sind in den Abbildungen B.1 und B.2 dargestellt. Diesen kann entnommen werden, dass in allen drei Messungen nach ca. 20 s die erzeugte CPU-Last weniger als um 5 % vom Sollwert abweicht.

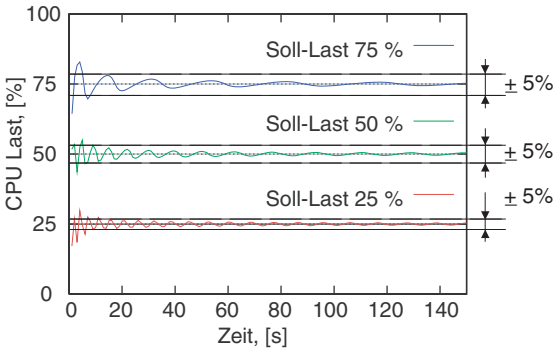

Abbildung B.1: Von Loadgen erzeugte CPU-Last bei einem einzelnen *Loadgen*-Prozess auf dem System

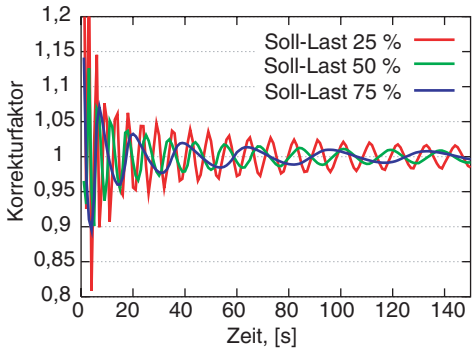

Abbildung B.2: Korrekturfaktor bei einem einzelnen *Loadgen*-Prozess auf dem System

In einem zweiten Szenario werden auf demselben Rechner zwei konkurrierende Instanzen

von Loadgen gestartet, wobei beide eine CPU-Last von 75% erzeugen sollen. Da im System nur eine CPU zur Verfügung steht, kann die vorgegebene Last von den *Loadgen*-Prozessen nicht erreicht werden. Somit muss jede Instanz bestrebt sein, die Dauer der aktiven Phase zu erhöhen, was sich dann im Wert des Korrekturfaktors widerspiegeln muss. Es wird erwartet, dass jeder Prozess etwa 50 % der CPU-Zeit erhält.

Das Ergebnis der Messung ist in den Abbildungen B.3 und B.4 dargestellt. Aus den dargestellten Abbildungen ist ersichtlich, dass die Güte des Systems für die in Kapitel 5.1.2.1 durchgeführten Untersuchungen ausreichend ist.

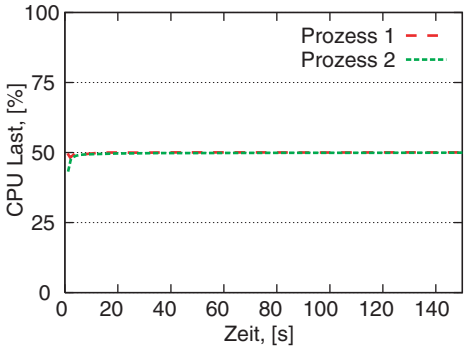

Abbildung B.3: Von *Loadgen* erzeugte CPU-Last bei zwei konkurrierenden *Loadgen*-Prozessen

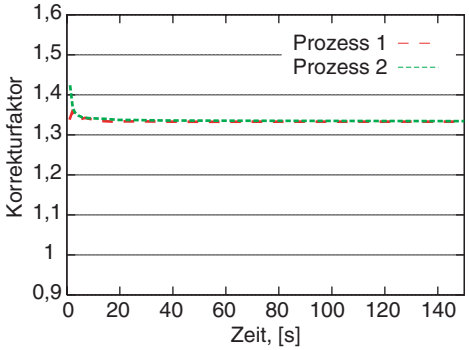

Abbildung B.4: Korrekturfaktor bei zwei konkurrierenden *Loadgen*-Prozessen

Anhang C

Die Unsicherheit des Task-Wechsels in Unix-Systemen

Die Unsicherheit der Freigabe der CPU für eine vordefinierte Zeit kann folgendermaßen anschaulich dargestellt werden:

In einem C-Programm wird in einer Schleife die CPU für eine Zeitspanne `delta` mit dem Systemaufruf *nanosleep* freigegeben. Initial wird `delta` auf eine Sekunde gesetzt. Bei jedem Durchlauf wird `delta` halbiert (gerundet auf Mikrosekunden). Dabei wird die Differenz der Systemzeit unmittelbar vor dem Systemaufruf *nanosleep* und unmittelbar danach gebildet. Tabelle C.1 zeigt die Ergebnisse, welche auf einem PC mit zwei Xeon 2,8 GHz Prozessoren, laufend unter *Linux 2.4.20*, sowie auf einer *Sun Enterprise 450* mit zwei SPARC-Prozessoren, laufend unter *Solaris 9*, erreicht werden. Die verwendeten Systeme waren bis auf übliche Hintergrund-Prozesse unbelastet. In der Tabelle sind die Ergebnisse eines einzelnen Durchlaufs dargestellt.

vorgegebene Zeit [µs]	**reale Zeit [µs]**		**absoluter Fehler [ms]**		**relativer Fehler**	
	Linux	Sun	Linux	Sun	Linux	Sun
1.000.000	1.005.044	1.003.830	5,044	3,830	$5,04 \cdot 10^{-3}$	$3,83 \cdot 10^{-3}$
500.000	509.851	509.358	9,851	9.358	$1,97 \cdot 10^{-2}$	$1,19 \cdot 10^{-2}$
250.000	259.941	259.845	9,941	9.845	$3,98 \cdot 10^{-2}$	$3,94 \cdot 10^{-2}$
125.000	139.935	129.723	14,935	4.723	$1,19 \cdot 10^{-1}$	$3,78 \cdot 10^{-2}$
62.5000	69.909	79.931	17,431	7,409	$2,79 \cdot 10^{-1}$	$1,19 \cdot 10^{-1}$
31.250	49.931	39.921	18,681	8,671	$5,98 \cdot 10^{-1}$	$2,77 \cdot 10^{-1}$
15.625	29.929	19.936	14,304	4,311	$9,16 \cdot 10^{-1}$	$2,76 \cdot 10^{-1}$
7.812	19.932	19.943	12,120	12,131	1,55	1,55
3.906	19.925	19.904	16,016	15.998	4,10	4,09

Fortsetzung auf der nächsten Seite

Fortsetzung, (siehe vorherige Seite)						
vorgegebene Zeit [µs]	**reale Zeit [µs]**		**absoluter Fehler [ms]**		**relativer Fehler**	
	Linux	Sun	Linux	Sun	Linux	Sun
1.953	19.929	19.929	17,976	17,976	9, 20	9, 20
976	19.930	19.932	18,954	18,956	19, 42	19, 42
488	19.929	19.932	19,441	19,444	39, 84	39, 84
244	19.929	19.924	19,685	19,680	80, 68	80, 66
122	19.930	19.931	19,808	19,809	162, 4	162, 4
61	19.939	19.934	19,878	19,873	325, 9	325, 8
30	19.921	19.923	19,891	19,893	663, 0	663, 1
15	19.928	19.933	19,913	19,918	1.3284	1.328
7	19.929	19.926	19,922	19,919	2.846	2.846
3	19.931	19.931	19,928	19,928	6.443	6.643
1	19.933	19.947	19,932	19,946	19.932	19.946

Tabelle C.1: Unterschied zwischen vorgegebenen und realen Wartezeiten eines Prozesses auf einem Linux-PC sowie auf einer Sun-Workstation

Literaturverzeichnis

[Act04] Acterna – Communications Test and Management Solutions. Product Home Page.
URL: `http://www.acterna.com/global/products/`, 2004

[Alm99a] G. Almes, S. Kalidindi, M. Zekauskas. *A One-way Delay Metric for IPPM.* RFC 2679, IETF, September 1999.

[Alm99b] G. Almes, S. Kalidindi, M. Zekauskas. *A One-way Packet Loss Metric for IPPM.* RFC 2680, IETF, September 1999.

[Alm99c] G. Almes, S. Kalidindi, M. Zekauskas. *A Round-trip Delay Metric for IPPM.* RFC 2681, IETF, September 1999.

[Bau03] A. Bauch, T. Heindorff. *Die SI-Basiseinheit „Sekunde“.* PTB-Mitteilungen, Heft 112(1), S. 291–298, 2003.

[Bie04] M. Bierwirth, J. Nehlmeyer. *Emulation von Störungen in hochperformanten IP-Netzen.* Bachelorarbeit, Universität Hannover, 2004.

[Bla98] S. Blake et al. *An Architecture for Differentiated Services.* RFC 2475, IETF, Dezember 1998.

[Bou88] S. Bourne. *Das Unix System V.* Addison Wesley Publishing Company, 1988.

[Bra94] R. Braden, D. Clark, S. Shenker. *Integrated Services in the Internet Architecture: an Overview.* RFC 1633, IETF, Juni 1994.

[Bra97] R. Braden, Ed., L. Zhang, S. Berson, S. Herzog, S. Jamin. *Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification.* RFC 2205, IETF, September 1997.

[Bri04] Brix Networks. Product Home Page.
URL: `http://www.brixnet.com/products/products_overview.html`, 2004

[Bro02] D. Brownell. *SAX2.* O'Reilly, 2002

[CAI04] CAIDA – Cooperative Association for Internet Data Analysis. Home Page. `URL: http://www.caida.org`, 2004.

[Car03] L. Carbone et al. *The Spectrum of Internet Performance.* In Proc.: Passive and Active Measurement Workshop (PAM2003), 2003.

[Cha01] J. Charzinski. *Worauf wir warten – Zur Dienstgüte in WWW-Verbindungen.* Workshop der ITG „IP in Telekommunikationsnetzen“, Bremen, 2001.

[Che97] A. Cherenson. *nslookup - Query Internet Name Servers Interactively.* Unix Manual Page. 1997

[Cis04a] Cisco Systems, Inc. *Cisco 7500 Series Routers: Quick Look Guide.* `URL: http://www.cisco.com/warp/public/cc/pd/rt/7500/prodlit/75sre_ov.pdf`, 2004.

[Cisc04b] Cisco Systems, Inc. *Cisco Catalyst 6500 Series: Supervisor Engine 720.* `URL: http://www.cisco.com/application/pdf/en/us/guest/products/ps708/c1650/ccmigration_09186a0080159856.pdf`, 2004.

[Cla99] J. Claus, G. Siegmund (Hrsgb). *Das ATM-Handbuch.* Hüthing Verlag, Heidelberg, 1999.

[Cla04] M. Claypool et al. *Inferring Queue Sizes in Access Networks by Active Measurement.* In Proc.: Passive and Active Measurement Workshop 2004 (PAM2004), 2004.

[Cra98] E. Crawley, R. Nair, B. Rajagopalan, H. Sandick. *A Framework for QoS-based Routing in the Internet.* RFC 2386, IETF, August 1998.

[Deg04] L. Degioanni, F. Risso. *PCAP – New Generation Dump File Format.* Internet Draft, IETF, März 2004.

[Dem02] C. Demichelis, P. Chimento. *IP Packet Delay Variation Metric for IP Performance Metrics (IPPM).* RFC 3393, IETF, September 2002.

[DFN96] DFN. *DFN-Betriebsstatistik.* DFN Mitteilungen, Heft 40, S. 31, März 1996.

[DFN04] DFN. *Betrieb und Nutzung des DFN.* DFN Mitteilungen, Heft 64, S. 31, März 2004.

[Dov01] C. Dovrolis, P. Ramanathan. *What do Packet Dispersion Techniques Measure ?* IEEE INFOCOM 2001, 2001.

[EGEE04] EGEE – Enabling Grids for E-science in Europe. Home Page. URL: `http://egee-intranet.web.cern.ch/egee-intranet/gateway.html`, 2004.

[Ets03] Y. Etsion, D. Tsafrir, D. G. Feitelson. *Effects of Clock Resolution on the Scheduling of Interactive and Soft Real-time Processes.* In Proc.: 2003 ACM SIGMETRICS Joint International Conference on Measurement and Modeling of Computer Systems, S. 172–183, San Diego, 2003.

[Fie99] R. Fielding et al. *Hypertext Transfer Protocol – HTTP/1.1.* RFC 2616, IETF, Juni 1999.

[Flo93] S. Floyd, Van Jacobson. *Random Early Detection Gateways for Congestion Avoidance.* IEEE/ACM Transactions on Networking, 1(4), S. 397–413, 1993.

[Fom04] M. Fomenkov, K. Keys, D. Moore, C. Claffy. *Longitudinal Study of Internet Traffic in 1998-2003.* Presented at: Winter International Symposium on Information and Communication Technologies, Mexico, 2004.

[GCC04] Gnu Compiler Collection. Online Dokumentation. URL: `http://gcc.gnu.org/onlinedocs/`, 2004.

[Gei00] K. Geihs, A. Puder, K. Romer. *MICO: An Open Source Implementation.* Morgan Kaufmann Publishers, 2000.

[Geo01] F. Georgatos et al. *Providing Active Measurements as a Regular Service for ISP's.* In Proc.: Passive and Active Measurement Workshop (PAM2001), RIPE NCC, 2001.

[Get95] Unix Manual Page. *gettimeofday, settimeofday – Get or Set the Date and Time.* 1995

[GGF04] Gloabal Grid Forum. Home Page. URL: `http://www.gridforum.org/`, 2004.

[Git02] W. Gitt. *Am Anfang war die Information.* Hänssler, 3. Aufl., 2002.

[Göh01] D. Göhr. *Bewertung der Dienstgüte von Audio- und Videodiensten.* Diplomarbeit, Technische Universiät Braunschweig, 2001.

[Grö04] R. Gröper. *Untersuchung von Methoden zur Synchronisierung entfernter Rechneruhren.* Bachelorarbeit, Universität Hannover, 2004.

[Grü98] L. Grüneberg. *Ein System zur Übertragung multimedialer Echtzeitdatenströme über Internet-Zugangsnetze.* Dissertation, Universität Hannover, 1998.

[Hac02] T. Hacker, B. Athey, B. Noble. *The End-to-End Performance Effects for Parallel TCP Sockets on a Lossy Wide-Area Network.* In Proc.: 16th IEEE-CS/ACM International Parallel and Distributed Processing Symposium (IPDPS), 2003.

[Hei01] K. Heidtmann et al. *Fehlertolerante Videokommunikation über verlustbehaftete Paketvermittlungsnetze.* Erschienen in: Killat U., Lamersdorf W. (Hrsg.), Kommunikation in Verteilten Systemen 2001, Springer, Berlin, 2001.

[HLR03] Norddeutscher Verbund für Hoch- und Höchstleistungsrechnen. Home Page.
`URL: http://www.hlrn.de`, 2003.

[Hol04] P. Holleczek. *Qualitätsmessungen im Gigabit-Wissenschaftsnetz.* DFN Mitteilungen, Heft 64, S. 22–23, März 2004.

[IEEE98] IEEE Standard 802.3u. *IEEE Information Technology - Telecommunications and Information Exchange between Systems - Local and Metropolitan Area Networks - MAC Parameters, Physical Layer, MAUs, and Repeater for 100 Mb/s Operation, Type 100BASE-T.* IEEE, 1998.

[IEEE99] IEEE Standard 802.3ab. *IEEE Information Technology - Telecommunications and Information Exchange between Systems - Local and Metropolitan Area Networks - Specific Requirements. Supplement to Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications - Physical Layer Parameters and Specifications for 1000 Mb/s Operation over 4-pair of Category 5 Balanced Copper Cabling, Type 1000BASE-T.* IEEE, 1999.

[IEEE02a] IEEE Standard 802.3. *IEEE Standard for Local and Metropolitan Area Networks— Part 3: CSMA/CD Access Method and Physical Layer Specifications Amendment: Media Access Control Parameters, Physical Layers and Management Parameters for Subscriber Access Networks.* IEEE, 2002.

[IEEE02b] IEEE Standard 802.3ae. *IEEE Standard for Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications-Media Access Control (MAC) Parameters, Pysical Layer and Management Parameters for 10 Gb/s Operation.* IEEE, 2002.

[IETF04] Internet Engineering Task Force. Home Page.
URL: http://www.ieft.org, 2004

[IPP04] IP Performance Working Group. Working Group Charter.
URL: http://www.ietf.org/html.charters/ippm-charter.html,
2004.

[ISO93] ISO/IEC 2382-1. *Information Technology – Vocabulary – Part 1: Fundamental Terms*. ISO/IEC, 1993.

[ITU93] ITU-T Recommendation I.120. *Integrated Services Digital Network (ISDN). General Structure. Integrated Services Digital Networks (ISDNs)* ITU-T Genf, März 1993.

[ITU94a] ITU-T Recommendation X.200. *Data Networks and Open System Communications. Open System Interconnection – Model and Notation. Information Technology – Open Systems Interconnection – Basic Reference Model: The Basic Model*. ITU-T Genf, Juli 1994.

[ITU94b] ITU-T Recommendation E.800. *Telephone Network and ISDN. Quality of Service, Network Management and Traffic Engineering. Terms and Definitions related to Quality of Service and Network Performance including Dependability* ITU-T Genf, August 1994.

[ITU96a] ITU-T Recommendation G.113. *Transmission Systems and Media. General Characteristics of International Telephone Connections and International Telephone Circuits. Transmission Impairments*. ITU-T Genf, Februar 1996.

[ITU96b] ITU-T Recommendation G.114. *Transmission Systems and Media. General Characteristics of International Telephone Connections and International Telephone Circuits. One-way Transmission Time*. ITU-T Genf, Februar 1996.

[ITU96c] ITU-T Recommendation G.729. *General Aspects of Digital Transmission. Coding of Speech at 8 kbit/s using Conjugate-structure Algebraic-code-excited Linear-prediction (CS-ACELP)*. ITU-T Genf, März 1996

[ITU97a] ITU-T Recommendation G.803. *Transmission Systems and Media, Digital Systems and Networks. Digital Transmission Systems – Digital Networks – General Aspects. Architecture of Transport Networks based on the Synchronous Digital Hierarchy (SDH)*. ITU-T Genf, Juni 1997.

[ITU97b] ITU-T Recommendation X.641. *Data Networks and Open System Communication. OSI Networking and System Aspects – Quality of Service. Information Technology - Quality of Service: Framework.* ITU-T Genf, Dezember 1997.

[ITU98] ITU-T Recommendation X.642. *Data Networks and Open System Communications. OSI Networking and System Aspects – Quality of Service. Information Technology – Quality of Service: Guide to Methods and Mechanisms.* ITU-T Genf, September 1998.

[Jac92] V. Jacobson, R. Braden, D. Borman. *TCP Extensions for High Performance*, RFC 1323, IETF, Mai 1992.

[Jai91] R. Jain. *The Art of Computer Systems Performance Analysis.* John Wiley & Sons Inc., New York, 1991.

[Jia04] H. Jiang, C. Dovrolis *The Origin of TCP Traffic Burstiness in Short-time Scale.* Technical report, Georgia Tech., 2004.

[Kan86] B. Kantor, P. Lapsley. *Network News Transfer Protocol: A Proposed Standard for the Stream-Based Transmission of News.* RFC 977, IETF, Februar 1986.

[Kle03] R. Kleineisel, I. Heller, S. Naegele-Jackson. *Messung von Echtzeitverhalten im G-WiN.* Pearl 2003, P. Holleczek, B. Vogel-Heuser (Hrsg.), Springer, 2003.

[Koo02] R. Koodli, R. Ravikanth. *One-way Loss Pattern Sample Metrics.* RFC 3357, IETF, August 2002.

[Kri01] R. Krienke. *Unix Sell Programmierung.* Hanser Fachbuchverlag, 2. Aufl., 2001.

[Kur02] J. Kurose, K. Ross. *Computer Networking.* Addison Wesley Publishing Company, 2002.

[Kus90] M. McKusick S. Leffler,M. Karels. *Das BSD UNIX Betriebssystem 4.3. Design und Implementierung.* Addison Wesley Publishing Company, 1990.

[Laa03] C. de Laat, E. Radius *High Performance Networking for Grid Applications.* In proc.: 9. annual conference of the Advanced School for Computing and Imaging, Heijen, Juni 2003

[Log01] D. Loguinov, H. Radha. *Measurement Study of Low-bitrate Internet Video Streaming*. In proc.: ACM SIGCOMM Internet Measurement Workshop, November 2001.

[Low02] B. Lowekamp et al. *A Hierarchy of Network Performance Characteristics for Grid Applications and Services*. Proposed Recommendation GFD-R.023, GGF, Mai 2004

[Luc01] M. J. Luckie, A. J. McGregor, H. Braun. *Towards Improving Packet Probing Techniques*. In Proc.: ACM SIGCOMM Internet Measurement Workshop 2001, San Francisco, CA, 2001.

[Mah99] J. Mahdavi, V. Paxson. *IPPM Metrics for Measurement Connectivity*. RFC 2678, IETF, September 1999.

[Mai01] G. Maiß. *DFN Videokonferenz*. Technischer Report, DFN, 2001.
`URL: http://www.dfn.de/uploaded/videokonferenz.pdf`

[Mar02] J. Martin, A. Nilsson. *On Service Level Agreements for IP Networks*. In proc.: IEEE INFOCOM, New York, NY, 2002.

[Mat01] M. Mathis, M. Allman *A Framework for Defining Empirical Bulk Transfer Capacity Metrics*. RFC 3148, IETF, Juli 2001.

[Mer77] L. Merz. *Grundkurs der Meßtechnik*. R. Oldenbourg Verlag, München, Wien, 5. Aufl., 1977.

[Mei04] Meinberg Funkuhren. Online Systembeschreibung.
`URL: http://www.meinberg.de/german/info/dcf77.htm`, 2004.

[MICO03] MICO Project. Home Page.
`URL: http://www.mico.org/`, 2003.

[Mil92] D. L. Mills. *Network Time Protocol (Version 3) Specification, Implementation and Analysis*. RFC 1305, IETF, März 1992.

[Mil96] D. L. Mills. *The Network Computer as Precision Timekeeper*. In Proc.: Precision Time and Time Interval (PTTI) Applications and Planing Meeting, Reston VA, 1996.

[Mlo96] *mlockall – Disable Paging for Calling Process*. Unix Manual Page. 1996

[Mor04] A. Morton, L. Ciavattone, G. Ramachandran, S.Shalunov, J. Perser. *Packet Reordering Metric for IPPM*. Internet Draft, IETF, August 2004.

[Moz04] Mozilla SVG Project. Home Page.
URL: http://www.mozilla.org/projects/svg/, 2004.

[Muu00] M. Muuss. The TTCP Program.
URL: http://ftp.arl.mil/ftp/pub/ttcp/, 2000.

[Nav03] J. Navratil, L. Cottrell. *ABwE: A Practical Approach to Available Bandwidth Estimation.* In Proc.: Passive and Active Measurement Workshop 2003 (PAM2003), 2003.

[Nic98] K. Nichols, S. Blake, F. Baker, D. Black. *Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers.* RFC 2474, IETF, Dezember 1998.

[NMWG04] Network Management Working Group, Home Page.
URL: http://www-didc.lbl.gov/NMWG/, 2004.

[OMG04] OMG. *CORBA/IIOP Specification, version 3.0.3*, Open Management Group.
URL: http://www.omg.org/technology/documents/formal/corba_iiop.htm, 2004.

[Ood97] A.P. Oodan, K.E. Ward, A.W. Mullee. *Quality of Service in Telecommunications.* The Institution of Electrical Engineers, London, 1997.

[OSI04] Open Source Initiative. *The BSD License.*
URL: http://opensource.org/licenses/bsd-license.html, 2004.

[Pas01] A. Pasztor, D. Veitch. *High Precision Active Probing for Internet Measurement.* In Proc.: INET 2001, Stockholm, Sweden, 2001.

[Pas02] A. Pasztor, D. Veitch. *PC Based Precision Timing Without GPS.* In Proc.: ACM SIGMETRICS, 2001.

[Pas03] A. Pasztor. *Accurate Active Measurement in the Internet and its Applications.* Ph. D. Thesis, University of Melbourn, Februar 2003.

[Pax97] V. Paxson. *Automated Packet Trace Analysis of TCP Implementations.* In proc.: ACM SIGCOMM, 1997.

[Pax98a] V. Paxson, G. Almes, J. Mahdavi, M. Mathis. *Framework for IP Performance Metrics.* RFC 2330, IETF, Mai 1998.

[Pax98b] V. Paxson. *On Calibrating Measurements of Packet Transit Times.* In Proc.: 1998 ACM SIGMETRICS Joint International Conference on Measurement and Modeling of Computer Systems, S. 11–21. ACM Press, 1998.

[Pos81a] J. Postel. *Internet Protocol. DARPA Internet Program Protocol Specification*. RFC 791, IETF, September 1981.

[Pos81b] J. Postel. *Internet Control Message Protocol. DARPA Internet Program Protocol Specification*. RFC 792, IETF, September 1981.

[Pos81c] J. Postel. *Transmission Control Protocol. DARPA Internet Program Protocol Specification*. RFC 793, IETF, September 1981.

[Pos82] J. Postel. *Simple Mail Transfer Protocol*. RFC 821, IETF, August 1982.

[Pos84] J. Postel. *Domain Name System Implementation Schedule – Revised*. RFC 921, IETF, Oktober 1984.

[PTB04] Physikalisch-Technische Bundesanstalt, Arbeitsgruppe 4.42, „Zeitübertragung“. Home Page.
`URL: http://www.ptb.de/de/org/4/44/442/index.htm`, 2004.

[Qua03] Quasar: Quality of Service Architecture, DFN-Projekt. Home Page.
`URL: http://www.ind.uni-stuttgart.de/Content/Quasar/`, 2003.

[Bur01] Lars Burgstahler et al. *Quality of Service Architekturen – Quasar, Milestone 1: Technology Overview.* `URL: http://www.ind.uni-stuttgart.de/Content/Quasar/publications/M1.pdf`

[Ran00] Standard C Library Functions Manual Page. *rand, srand, rand_r - simple random-number generator*. 2000.

[RIPE] RIPE Network Coordination Centre. Home Page.
`URL: http://www.ripe.net/`, 2004.

[Rob95] D. Robinson, C. Lonvick. *The Common Gateway Interface, version 1.1.*
`URL: http://www.w3.org/CGI/`.

[Ros02] J. Rosenberg et al. *SIP: Session Initiation Protocol*. RFC 3261, IETF, Juni 2002.

[RUD04] RUDE Project. Home Page at Sourceforge.
`URL: http://rude.sourceforge.net/`, 2004.

[Sch03] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson. *RTP: A Transport Protocol for Real-Time Applications*. RFC 3550, IETF, Juli 2003.

[SEQ04] SEQUIN – SErvice QUality across Independently managed Networks. Project Home Page.
`URL: http://archive.dante.net/sequin/`, 2004.

[Set02] Unix Manual Page. *sched_setscheduler, sched_getscheduler – Set and Get Scheduling Algorithm/Parameters*. 2002

[Sha48] C. E. Shanon. *A Mathematical Theory of Communication*. Bell System Technical Journal, 27, S. 379–423, 623–656, 1948.

[Sha04] S. Shalunov, A. Karp, J. W. Boote, M.Zekauskas. *A One-way Active Measurement Protocol (OWAMP)*. Internet Draft, IETF, August 2004.

[SI-04] Physikalisch-Technische Bundesanstalt, Braunschweig. *Das internationale Einheitensystem (SI)*.
`URL: http://www.ptb.de/de/wegweiser/einheiten/si/index.html`, 2004.

[Sie99a] G. Siegmund. *Technik der Netze*. Hüthing Verlag Heidelberg, 4. Aufl., 1999.

[Sie00] E. Siemens. *Realisierung und Bewertung eines QoS-Dienstes im Campus-Netz*. Diplomarbeit, Universität Hannover, Mai 2000.

[Sie01a] E. Siemens, H. Pralle. *Kopplung von TK-Anlagen über ein IP-Netz*. Erschienen in: J. v. Knop, W. Haverkamp (Hrsg.), Innovative Anwendungen in Kommunikationsnetzen. 15. DFN-Arbeitstagung über Kommunikationsnetze, Düssldorf, S. 55-61, Köllen Verlag, Berlin, 2001:

[Sie03] E. Siemens, C. Grimm, S. Piger. *Verkehrsprofile echtzeitkritischer Sprachdienste in IP-Netzen*. In Proc.: Kommunikation in Verteilten Systemen (KiVS2003), 2003.

[Sie04a] E. Siemens, S. Piger, C. Grimm, M. Fromme. *LTest – A Tool for Distributed Network Performance Measurement*. In Proc.: 2004 IEEE Consumer Communications and Networking Conference (CCNC2004), 2004.

[Sie04b] E. Siemens. LTest. Project Home Page.
`URL: http://www.rrzn.uni-hannover.de/ltest.html`, 2004.

[Sil98] A. Silberschatz, P. Galvin. *Operating System Concepts*. Addison Wesley Longnam, Inc, 5. Aufl., 1998

[Smo03] V. Smotlacha. *One-way Delay Measurement using NTP*. Presented at: Terena Networking Conference, 2003.

[Sol92] K. Sollins. *The TFTP Protocol (Revision 2)*. RFC 1350, IETF, Juli 1992.

[SSL96] Netscape Communication Corporation. *Secure Socket Layer*. Protocol Specification (Version 3).
`URL: http://wp.netscape.com/eng/ssl3/`, 1996.

[Ste92] W. R. Stevens. *Advanced Programing in the Unix Environment*. Addison Wesley Publishing Company, 1992.

[Ste94a] W. R. Stevens. *TCP/IP Illustrated, Volume 1. The Protocols*. Addison Wesley Publishing Company, 1994.

[Ste94b] W. R. Stevens. *TCP/IP Illustrated, Volume 3. TCP for Transactions, HTTP, NNTP, and the UNIX Domain Protocols*. Addison Wesley Publishing Company, 1994.

[Ste97a] W. R. Stevens. *TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms*. RFC 2001, IETF, January 1997.

[Ste97b] W. R. Stevens. *Unix Network Programming*. Prentice Hall PTR, 1997.

[Str92] W. Strayer, B. Dempsey, A. Wewver. *XTP: The Xpress Transfer Protocol*. Addison-Wesley Publishing Company, 1992.

[Str00] B. Stroustrup. *Die C++ Programmiersprache*. Addison Wesley Publishing Company, 2000.

[Sub01] L. Subramanian V. N. Padmanabhan, R. H. Katz *Geographic Properties of Internet Routing: Analysis and Implications.* USENIX Annual Technical Conference, Monterey, CA, 2002

[T1.523] Alliance for Telecommunications Industry Solutions. *American National Standard for Telecommunications - Telecom Glossary*. ANS T1.523-2001, 2000. American National Standard, 2001.

[Tan95] A. Tanenbaum. *Moderne Betriebssysteme*. Hanser, 2. Aufl., 1995.

[Tan01] A. Tanenbaum. *Modern Operating Systems*. Prentice-Hall, Inc., second edition., 2001.

[Tcp04] Tcpdump Project. Home Page.
`URL: http://www.tcpdump.org/`, 2004

[Tir03] A. Tirumala, L. Cottrell, T. Dunigan. *Measuring End-to-end Bandwidth with Iperf using Web100*. Presentation at Internet2 Spring Members Meeting, 2003.

[Tir04] A. Tirumala et al. *Iperf - The TCP/UDP Bandwidth Measurement Tool.* URL: http://dast.nlanr.net/Projects/Iperf, 2004.

[Ubi01] S. Ubik, V. Smotlacha, S. Saaristo, J. Laine. *Low-cost Precise QoS Measurement Tool.* Technical report, CESNET, Prague and Tampere University and Soon Communications, 2001.

[UTC] Bureau International des Poids et Mesures (BIPM): *Approximation to UTC.* URL: http://bipm.org/en/scientific/tai/time_server.html, 2004.

[Var04] G. Varghese, C. Estan. *The Measurement Manifesto.* SIGCOMM Computer Communicatoins Rev., 34(1) S. 9–14, 2004.

[Vay04] I. Vayner. *Visualisierung simulierter Datenströme in IP-Netzen.* Diplomarbeit, Universiät Hannover, 2004.

[W3C04] World Wide Web Consortium - W3C, Home Page. URL: http://www.w3.org/, 2004.

[W3Ca] World Wide Web Consortium. Extensible MarkupLanguage (XML), Version 1.1. URL: http://www.w3.org/TR/xml11/.

[W3Cb] World Wide Web Consortium. Scalable Vector Graphics, Version 1.1. URL: http://www.w3.org/Graphics/SVG/.

[W3Cc] World Wide Web Consortium. Document Object Model (DOM). URL: http://www.w3.org/DOM/, 2004.

[W3Cd] World Wide Web Consortium. SOAP Version 1.2 Part 0: Primer, W3C Reccomendation. URL: http://www.w3.org/TR/2003/REC-soap12-part0-20030624/, 2003.

[Web04] Web100 Project. Home Page. URL: http://www.web100.org/, 2004.

[Win04] WiN Labor, Erlangen. Home Page. URL: http://www.win-labor.dfn.de/, 2004.

[XML04] The XML C parser and toolkit of Gnome. Project Home Page. URL: http://www.xmlsoft.org/, 2004.

[Ylo04] T. Ylonen, C. Lonvick. *SSH Protocol Architecture*. Internet Draft, IETF, Oktober 2004.

[Yod99] V. Yodaiken. *The RTLinux Manifesto*. Technical Report. `URL: http://www.fsmlabs.com/articles/archive/manifesto.html`, 1999.

[Zse02] T. Zseby, F. Schreiner. *QoS Monitoring and Measurement Benchmarking*. Next Generation Networks Initiative. Project Deliverable D1, 2002.

Tabellarischer Lebenslauf

Eduard Siemens

Persönliche Daten

05.02.1969	geboren in Uljanowskoje, Kasachstan als Sohn von Gerhard Siemens und seiner Ehefrau Frieda Siemens, geb. Epp
seit 15.12.1992	verheiratet mit Tatjana Siemens, geb. Kazantsev
11.11.1993	Geburt des Sohnes Markus
09.09.1995	Geburt des Sohnes Thomas
24.01.2001	Geburt der Tochter Anna-Liese
22.09.2004	Geburt des Sohnes Ben Philipp

Schule und Ausbildung

1976-1986	Allgemeinbildende Schule in Uljanowskoje, Kasachstan
1987-1989	Wehrdienst, Kasachstan
1986-1991	Fernstudium Elektroenergetik, Novosibirsk
1995-2000	Universität Hannover, Studium Elektrotechnik mit dem Schwerpunkt Kommunikationsnetze
Mai 2000	Abschluss des Studiums mit der Diplomhauptprüfung

Beruflicher Werdegang

1986-1987	Fernmeldeamt Elektromonteur, Installateur
1989-1991	Fernmeldeamt Elektromonteur, Wartung und Betrieb von Alarmanlagen
1991-1993	Bund Evangelisch-Freikirchlicher Gemeinden, Kasachstan. Karitative Tätigkeit
1993-1994	Technisch-Wissenschaftlicher Betrieb „KSU“, Novosibirsk. Projektingenieur
2000-2005	Universität Hannover, Lehrgebiet Rechnernetze und Verteilte Systeme. Wissenschaftlicher Mitarbeiter. Mitarbeiter und Projektleiter von Projekten im Bereich Forschung und Entwicklung, Kommunikationssysteme