Methoden zur Leistungsbewertung von Internet-Zugangspunkten

Inaugural-Dissertation
zur Erlangung des Doktorgrades
der Mathematisch-Naturwissenschaftlichen Fakultät
der Universität zu Köln

vorgelegt von
Martin Horneffer
aus Berlin

Köln
2000

Berichterstatter:	Prof. Dr. R. Schrader Prof. Dr. E. Speckenmeyer
Tag der mündlichen Prüfung:	11. Juli 2000

Dank

Ich danke Professor Dr. Rainer Schrader und Dr. Wolfgang Trier für die Betreuung und Unterstützung dieser Arbeit. Sie haben mir viele gute Hinweise gegeben und mich bei meinen Vorhaben immer wieder in die richtige Richtung gelenkt. Dr. Trier hat mir etliche Reisen zu internationalen Tagungen ermöglicht, auf denen ich Anschluss an die weltweit verteilte Gemeinde der Internet-Netzforscher finden konnte. Professor Dr. Evi Nemeth hat mich auf einigen dieser Reisen unterstützt, indem sie mich als „MBone-Slave“ beschäftigt und dadurch zur Reisefinanzierung beigetragen hat, und, indem sie mich mit interessanten Leuten in Kontakt gebracht hat. Für beides bin ich ihr sehr dankbar. Den vielen Kollegen aus unserem Rechenzentrum einschließlich des ehemaligen Kollegen Axel Clauberg, die mir bei zahlreichen technischen Fragen geholfen haben, gilt mein herzlichster Dank. Nur durch die besondere Infrastruktur unseres Rechenzentrums und das offene, freundliche Arbeitsklima ist die vorliegende Arbeit überhaupt möglich geworden. Auch mussten sie mehr als einmal die Beschwerden und Beschuldigungen der „Opfer“ meiner Messungen entgegennehmen und an mich weiterleiten. Nicole Lyscom und Roland Franzen danke ich für ihre überaus wertvolle Hilfe bei der englischsprachigen Formulierung der Teilpublikationen. Ein besonderer Dank gilt auch Michael Rüsel von der Firma NetCologne GmbH. Durch seine Hilfe konnte ich meine Messmethoden auf einen zusätzlichen Internet-Zugangspunkt anwenden, was die Ergebnisse ungleich interessanter werden ließ. Und schließlich danke ich Isolde Gerlach dafür, dass sie mir privat den Rücken frei gehalten und mich unterstützt hat, auch als sie mich wegen regelmäßiger Nachtschichten kaum zu Gesicht bekam.

Inhaltsverzeichnis

Kapitel 1

Einleitung

In den letzten Jahren hat das Internet ein enormes Wachstum erlebt und sich in kürzester Zeit zu einem festen Bestandteil der Infrastruktur unserer Gesellschaft entwickelt. Das Wachstum und die Dynamik des Internet sind bislang ungebrochen — immer mehr Menschen und Firmen suchen Zugang zum Internet und wickeln immer größere Teile ihrer Geschäfte darüber ab. Gleichzeitig etablieren sich immer mehr Anbieter auf dem Markt, die Kunden auf verschiedenste Art und Weise Zugang zum Internet bieten — mit den verschiedensten Zugangstechniken und mindestens genauso vielfältigen Vertrags- und Geschäftsmodellen. Aufgrund dieses raschen Wachstums und Wandels, dessen Entwicklung kaum überschaubar ist, besteht Bedarf an Verfahren, die Leistungsfähigkeit von Internet-Zugangspunkten objektiv und vergleichbar zu bestimmen.

So ist neben der Frage der verwendeten Zugangstechnik, also der Verbindung zwischen Kunden und Anbieter, auch die Einbindung in das weltweite Internet für die Leistungsfähigkeit eines Zugangspunktes von entscheidender Bedeutung. Die Qualität der Dienste verschiedener Anbieter kann sich dabei sehr unterscheiden. Betreibt ein Anbieter leistungsfähige und ausgewogene Verbindungen zu den anderen Teilen des Internet, so hat er dadurch in der Regel zusätzliche Aufwendungen gegenüber einem Anbieter, der allein auf niedrige Kosten achtet. Es kann für einen Benutzer aber durchaus von Vorteil sein, bei einem vermeintlich „teureren“ Anbieter Kunde zu werden, wenn dort das Preis-/Leistungsverhältnis für seinen spezifischen Bedarf besser ist oder wenn bestimmte Anwendungen erst durch eine hochwertige Internet-Anbindung möglich werden.

Dies wurde auch außerhalb der engeren Fachwelt bereits als Problem erkannt [52, 53]. Allerdings gibt es für die Leistungsfähigkeit der Internet-Anbindung bislang kein allgemein anerkanntes Maß, das einen direkten Vergleich der Gesamtleistung verschiedener Zugangspunkte erlauben würde und über das Testen einiger willkürlich ausgewählter Internet-Verbindungen hinausgeht.

1.1 Ziel der Arbeit

Ziel der vorliegenden Arbeit ist es, Maße und Methoden zu entwickeln und zu untersuchen, mit denen die Einbindung eines Internet-Zugangspunktes in das Internet leistungsmäßig bewertet werden kann.

Dazu sollen geeignete *Größen* definiert und praktikable *Messmethoden* entwickelt und untersucht werden. Um die praktische Bedeutung dieser Größen für den Endbenutzer sinnvoll einschätzen zu können, sollen die *Auswirkungen* der messbaren Größen auf die Performance von typischen Internet-Anwendungen untersucht werden.

Wichtige Kriterien für das zu entwickelnde Instrumentarium sollen sein: *Objektivität*, das heißt Unabhängigkeit von willkürlich ausgewählten Messstrecken, *statistische Aussagekraft* der Ergebnisse, sowie die *praktische Anwendbarkeit*.

1.2 Begriffsbestimmungen und Modelle

In diesem Kapitel werden zunächst einige Begriffe definiert und erläutert sowie einige Modelle vorgestellt, die von grundlegender Bedeutung für die vorliegende Arbeit sind und die im folgenden häufiger verwendet werden. Mit dem Pfeil ↑ gekennzeichnete Begriffe werden an anderer Stelle in diesem Kapitel definiert oder erläutert, gegebenenfalls weiter unten.

Protokoll Ein **Protokoll** („protocol“) ist eine Menge von Regeln, die den Informationsfluss in Kommunikationssystemen steuern [30].

Host Unter einem **Host** wird im folgenden eine elektronische Datenverarbeitungsanlage verstanden, die von ihrer Ausstattung her dazu geeignet ist, mit anderen Hosts unter Verwendung der ↑TCP/IP-Protokolle Daten auszutauschen. Beispiele sind Computer im Sinne von Arbeitsplatzrechnern oder auch Servern, aber auch bestimmte Drucker und andere Peripheriegeräte sowie ↑Router.

Netz Ein **Netz** („net“, „network“) ist eine Menge von Punkten, die über Kommunikationsleitungen miteinander verbunden sind [19]. Dies ist eine sehr allgemeine Definition. Sie lässt sich von Fall zu Fall konkretisieren, indem das Netz beispielsweise auf die Verwendung einer bestimmten Technik beschränkt wird. So ist ein **Ethernet-Netz** eine Menge von Geräten, die mittels Ethernet-Technik [33] verbunden sind.

Ein **IP-Netz** ist eine Menge von Hosts, die so miteinander verbunden sind, dass sie unter Verwendung des Protokolls ↑IP Daten austauschen können. Es lässt sich in erster Näherung gut durch einen zusammenhängenden

Graphen mit vielen Knoten und beliebigen Kanten zwischen den Knoten darstellen. Jeder Knoten steht dabei für einen Host, jede Kante für eine Kommunikationsleitung. In IP-Netzen unterscheidet man verschiedene Klassen von Hosts und verwendet in Abhängigkeit der Klasse bestimmte Bezeichnungen. Das Hauptkriterium dafür ist die Zahl n der Kanten eines Knotens:

$n = 0$ Dies ist ein **isolierter Host** ohne Verbindung zu anderen Hosts.

$n = 1$ Ein **einfacher Host** hat genau einen Anschluss an ein IP-Netz. Beispiele sind normale Arbeitsplatzrechner, PC oder auch Server-Rechner.

$n > 1$ Hier muss man zwei Fälle unterscheiden und zwar in Abhängigkeit davon, ob auf dem Host ↑Routing betrieben wird. Ist dies der Fall, so wird der Host **Router** genannt, andernfalls spricht man von einem **Multi-Homed Host**.

Als Router lässt sich normalerweise jeder Computer mit vollständigem ↑IP-Protokoll und mehreren Netz-Interfaces einsetzen. In der Regel verwendet man dafür jedoch spezielle Geräte, die von Ihrer Hardware und von ihrem Betriebssystem her auf diese Aufgabe spezialisiert sind.

Ein Multi-Homed Host ist in der Regel ein Server- oder Arbeitsplatzrechner mit besonderen Anforderungen, etwa mit großem Bedarf an Bandbreite oder Zuverlässigkeit, der von einem Netz-Interface alleine nicht gedeckt werden kann; oder mit der Anforderung, von einem Rechner aus auf verschiedene ↑Internet-Zugangspunkte zugreifen zu können.

Internet Das **Internet** ist ein weltweiter Zusammenschluss von ↑IP-Netzen. Die Organisation **IETF** (Internet Engineering Task Force [35]) entwickelt, beschreibt und definiert die im Internet verwendeten Protokolle und die Organisation **ICANN** (Internet Corporation for Assigned Names and Numbers [26]) organisiert die eindeutige Vergabe von ↑Adressen, Namen und Protokollbezeichnern im Internet.

Die IETF veröffentlicht die Internet-Protokolle sowie grundlegende Regeln zum Betrieb des Internet in einer Dokumenten-Reihe mit dem Namen **RFC** („Request For Comments“). Jedes Dokument dieser Reihe bekommt eine eindeutige, fortlaufende Nummer, ist öffentlich und steht auf den Servern der IETF jedermann kostenlos zu Verfügung. Ein einmal veröffentlichtes RFC-Dokument kann nachträglich nicht mehr verändert werden. Es kann aber in bestimmten Fällen von neueren RFC-Dokumenten ergänzt oder abgelöst werden.

Die Gesamtheit der im Internet eingesetzten Protokolle wird oft kurz als **TCP/IP** bezeichnet, da ↑IP und ↑TCP die wichtigsten und bekanntesten Protokolle des Internet sind.

Dienst Ein **Dienst** („service“) ist eine Menge an Funktionen, die eine Schicht („layer“) einem Benutzer („client“) anbietet [30].

Paket Ein **Paket** ist eine Menge von Bit, einschließlich solcher für Daten- und für Protokollinformationen, die in einem Verbund über ein Paket-orientiertes Netz übertragen wird [30].

Adresse Eine **Adresse** ist ein eindeutiger Bezeichner für den Ort bestimmter Daten oder für ein intelligentes Gerät [30].

Eine **IP-Adresse** ist nach der heute üblichen Version 4 des IP-Protokolls eine 32 Bit umfassende Zahl, die eindeutig einen Host kennzeichnet. Die eindeutige Vergabe von IP-Adressen wird von der ICANN beziehungsweise von von ihr beauftragten Institutionen geregelt. Die übliche Schreibweise für IP-Adressen sieht vor, die vier Byte der 32-Bit-Zahl einzeln dezimal auszuschreiben und mit Punkten zu trennen. Beispiel: 134.95.129.25

Schichtenmodell Dem Internet liegt ein dem ISO/OSI-Referenzmodell [72] ähnliches, aus mehreren *Schichten* bestehendes Modell zu Grunde. Dabei liegen verschiedene Schichten übereinander, denen jeweils bestimmte Funktionen zugeordnet sind, die mit Hilfe geeigneter Protokolle erbracht werden. Jede Schicht nutzt Dienste der nächst tieferen Schicht, um mit einem bestimmten Mehrwert ihrerseits Dienste für die nächst höhere Schicht zu erbringen.

Während das ISO/OSI-Referenzmodell sieben Schichten definiert, unterscheidet man in IP-Netzen im wesentlichen vier Schichten, siehe Abbildung 1.1 [77].

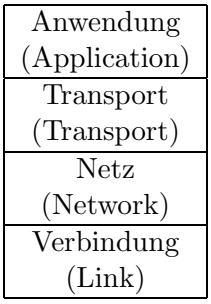

Abbildung 1.1: Die vier Schichten des TCP/IP-Protokollstapels.

- Die **Verbindungsschicht** („link layer“, „data-link layer“ oder auch „network interface layer“) sorgt für die Übertragung von Daten zwischen Hosts, die an ein gemeinsames Übertragungsmedium angeschlossen sind. Ihr Endpunkt ist typischerweise ein Gerätetreiber des Be-

triebssystems. Beispiele für Protokolle dieser Schicht sind Ethernet [33], FDDI [6, 30] oder auch PPP [76].

- Die **Netzschicht** („network layer“ oder „internet layer“) hat das Weiterleiten und Verteilen von Paketen im Netz zur Aufgabe, auch ↑Routing genannt. Die Protokolle **IP** (Internet Protocol [65]), **ICMP** (Internet Control Message Protocol [66]) und **IGMP** (Internet Group Management Protocol [20]) sind in dieser Schicht zugeordnet.

  Die Netzschicht hat in IP-Netzen lediglich die Aufgabe, Pakete von einem Host A zu einem Host B zu leiten. Sie stellt nicht sicher, dass ein Paket tatsächlich am Ziel ankommt, oder dass es innerhalb einer bestimmten Zeit ankommt. Das ist gegebenenfalls die Aufgabe höherer Schichten.

- Die **Transportschicht** stellt einen kontrollierten Datenfluss zwischen zwei Hosts im Internet zu Verfügung. Im Internet kommen im wesentlichen zwei Transport-Protokolle zum Einsatz, die sich grundlegend unterscheiden: ↑TCP zum gesicherten Transport beliebiger Datenströme und ↑UDP zum einfach, ungesicherten Transport einzelner Datensätze.

- Die **Anwendungsschicht** beinhaltet alle Einzelheiten der Anwendungen. Es gibt im Internet eine unüberschaubare Vielfalt an Diensten und Anwendungen und täglich kommen neue hinzu. Allerdings gibt es immer bestimmte Protokolle, auf denen die am häufigsten eingesetzten Anwendungen basieren, zum Beispiel **HTTP** (HyperText Transfer Protocol [10]) für das WWW (World Wide Web), **FTP** (File Transfer Protocol) für die Übertragung von Dateien, **SMTP** (Simple Mail Transfer Protocol, [68]) für den Austausch von E-Mail, **NNTP** (Net News Transfer Protocol [68]) für die Übertragung von Net-News, etc.

Anders als im ISO/OSI-Referenzmodell ist die Zuordnung verschiedener Protokolle und Funktionen zu den verschiedenen Schichten im Internet nicht ganz konsequent durchgeführt. So sind etwa das ICMP- und das IGMP-Protokoll zwischen der Netz- und der Transportschicht einzuordnen. Sie liegen über IP, erbringen aber Dienste sowohl für die Transportschicht (etwa für UDP), als auch für die Netzschicht selber. Des weiteren gibt es Anwendungen mit besonderen Anforderungen an die Transportschicht, die weder von ↑TCP noch von ↑UDP erfüllt werden. So gibt es zum Beispiel **RTP** (Real-Time Transport Protocol [74]), das üblicherweise in der Applikation selber implementiert wird und auf UDP aufbaut, eigentlich aber Aufgaben der Transportschicht erfüllt.

Routing Sowohl der Zusammenschluss von lokalen Netzen wie Ethernet- oder FDDI-Netzen zu IP-Netzen, als auch der Zusammenschluss von vielen IP-Netzen zum weltweiten Internet wird dadurch ermöglicht, dass Router

IP-Pakete, die sie auf einem Anschluss empfangen, über einen anderen Anschluss wieder ausgegeben und dabei ihrem Ziel näher bringen. Dies wird **Routing** oder auch **IP-forwarding** genannt.

Um diese Aufgabe erfüllen zu können, hat ein Router eine sogenannte **Routing-Tabelle**, die ihm anhand der Ziel-IP-Adresse[1] sagt, an welchen Nachbarn er ein IP-Paket weiterleiten muss. Routing-Tabellen werden entweder für jeden Router statisch festgelegt oder von speziellen Routing-Protokollen automatisch angelegt und aktualisiert. Ein Router kann mit diesem Verfahrens zustandslos arbeiten und ohne zusätzliche Informationen über bestehende ↑IP-Verbindungen jedes einzelne IP-Paket in die richtige Richtung weiterleiten.

TCP Das Protokoll **TCP** (Transmission Control Protocol [67]) stellt die gesicherte Übertragung eines Datenstroms zwischen zwei Anwendungen auf IP-Hosts zu Verfügung. Damit können beliebig viele Daten vom Sender zum Empfänger übertragen werden und TCP sorgt dafür, dass die Daten fehlerfrei und in der richtigen Reihenfolge ankommen. Mit Hilfe einer ↑Portnummer werden verschiedene Anwendungen auf einem Host unterschieden.

Eine herausragende Eigenschaft von TCP ist es, einerseits den Datendurchsatz laufend zu optimieren und andererseits Überlastungserscheinungen im Netz zu erkennen und den Durchsatz jeder einzelnen Übertragung so zu verringern, dass das Netz als ganzes nicht zusammenbricht, sondern weiterhin so gut wie möglich arbeitet. Die dafür eingesetzten Verfahren werden in Kapitel 7.3 kurz beschrieben.

UDP Das Protokoll **UDP** (User Datagram Protocol [64]) auf der anderen Seite stellt einen wesentlich einfacheren Dienst zu Verfügung. Es werden nur einzelne Datensätze übertragen, die eine bestimmte Größe nicht überschreiten dürfen. Wie auf der IP-Ebene wird auch hier nicht garantiert, dass ein Paket überhaupt beim Empfänger ankommt oder dass mehrere Pakete in der richtigen Reihenfolge ankommen. Der Mehrwert von UDP als Transportprotokoll beschränkt sich darauf,

- mittels einer einfachen Prüfsumme die Integrität der Datensätze zu sichern und
- mittels einer ↑Portnummer mehrere Anwendungen auf einem Host zu unterschieden.

[1] Die Ziel-IP-Adresse ist Bestandteil der Protokollinformationen jeden IP-Pakets. Andere Bestandteile sind beispielsweise die IP-Adresse des Absenders und die Bezeichnung des Transportprotokolls.

Portnummer Die Transportprotokolle TCP und UDP verwenden eine sogenannte **Portnummer**, um sowohl auf dem Sende- als auch Empfangshost mehrere Anwendungen unterscheiden zu können. Dabei handelt es sich um ganze Zahlen von 1 bis 65535, die in jedem TCP- und UDP-Paket enthalten sind. Sie haben entweder einen festen, typischerweise von der ICANN [26] definierten Wert, oder einen dynamisch vom jeweiligen Betriebssystem vergebenen Wert. Weitere Erläuterungen dazu finden sich in Kapitel 4.3.2.

Netzmodell des Internet Wie bereits angedeutet lassen sich ↑IP-Netze als Graph modellieren. Wie den für Hosts stehenden Knoten lassen sich auch den für Kommunikationsleitungen stehenden Kanten dieses Modells charakteristische Eigenschaften zuordnen. Beispiele sind die ↑Bandbreite und ↑Paketlaufzeit. In bestimmten Fällen werden über ein Übertragungsmedium mehr als zwei Hosts miteinander verbunden, man spricht dann von einem „shared medium“. Ein typisches Beispiel dafür ist Ethernet. Im Modell kann dies berücksichtigt werden, indem das Ethernet durch viele Kanten dargestellt wird, die eine Vollvermaschung aller Knoten bilden, die für die am Medium angeschlossenen Hosts stehen. Die genauen Eigenschaften der Kanten sind dann allerdings nicht mehr unabhängig voneinander. Das Modell ist also keineswegs perfekt und nicht unbedingt geeignet, alle Details des Internets zu beschreiben. Abgesehen davon, dass ein perfektes Modell des Internet sowieso illusorisch ist [63], genügt das einfache Modell aber für die Betrachtungen dieser Arbeit.

Bandbreite Die **Bandbreite** einer Kommunikationsleitung ist ihre mögliche Datenübertragungsrate, gemessen in Bit pro Sekunde.

Eine *Kommunikationsleitung* ist dann *schmalbandig*, wenn ihre Bandbreite „klein“ ist. Eine konkretere Definition der **Schmalbandigkeit** ist nicht allgemein möglich, da sie vom Betrachter und vom Bedarf abhängt und da sich die Vorstellungen „normaler“ Bandbreite im Laufe der Zeit ändern. Für die Internet-Zugangsleitung einer Privatperson würde man Schmalbandigkeit heutzutage etwa bei maximal 128 Kbit/s annehmen. Modem- und ISDN-Zugängen wären demnach schmalbandig, xDSL- und Kabelnetz-Zugänge breitbandig. Für Universitäten und Großforschungseinrichtungen zog der DFN-Verein die Grenze bei 2 Mbit/s, als er mit dem „Breitband-Wissenschaftsnetz“ erstmalig Internet-Zugangsleitungen mit mehr als 2 Mbit/s anbot.

Eine *Anwendung* ist schmalbandig, wenn die von Ihr beanspruchte Bandbreite klein ist im Vergleich zur Bandbreite der zu Verfügung stehenden Internet-Zugangsleitung.

Paketlaufzeit Die **Paketlaufzeit** („delay“) ist die Zeit, die ein Paket benötigt, um von einem Punkt eines Netzes zu einem anderen zu gelangen.

Genaue Definitionen für Paketlaufzeiten im Internet finden sich in Kapitel 3.2. Für das hier betrachtete Netzmodell soll gelten, dass eine Kommunikationsleitung immer genau ein Bit zu einer Zeit überträgt und alle Bit eines Pakets hintereinander. Wenn τ_S die ↑*Sendezeit* des Pakets auf der Leitung und τ_A die ↑*Ausbreitungszeit* der Leitung ist, dann gilt für die *Paketlaufzeit einer Leitung* τ_L:

$$\tau_L = \tau_S + \tau_A \tag{1.1}$$

Die **Sendezeit** τ_S eines Pakets auf einer Leitung ist die Zeit, die benötigt wird, das Paket auf die Leitung zu senden, also vom Beginn des Sendens des ersten Bit bis zum Ende des Sendens des letzten Bit des Pakets. Ist B die Bandbreite einer Leitung und L die Länge eines Pakets, dann gilt:

$$\tau_S = L/B \tag{1.2}$$

Die **Ausbreitungszeit** τ_A einer Leitung ist die Zeit, die ein Bit zum Durchlaufen der Leitung vom einen bis zum anderen Ende braucht, also die Zeit vom Absenden eines Bit am Sender bis zum Empfang des Bit am Empfänger. Ist l die physikalische Länge einer Leitung und c ihre Signalausbreitungsgeschwindigkeit, so gilt:

$$\tau_A = l/c \tag{1.3}$$

Mit 1.2 und 1.3 lässt sich 1.1 auch schreiben als:

$$\tau_L = L/B + l/c \tag{1.4}$$

Routermodell Ein einfaches Modell für die Vorgänge in einem Router[2] ist in Abbildung 1.2 dargestellt. Jeder Verbindung (oder Kante im Netzmodell) entsprechen dabei ein Eingang und ein Ausgang. Über die Eingänge gelangen Pakete aus beliebigen Richtungen in den Router. Dann entscheidet ein zentraler Prozessor für jedes IP-Paket in Abhängigkeit der Ziel-IP-Adresse, in welche Richtung, also zu welchem Ausgang, es weitergeleitet werden muss. Vor dem Ausgang muss das Paket gegebenenfalls noch in einer Warteschlange zwischengepuffert werden.

Nun kann es leicht passieren, dass durch mehrere Eingänge mehr Pakete für einen bestimmten Ausgang in den Router gelangen, als dieser Ausgang entsprechend seiner Bandbreite weiterleiten kann. Dadurch füllt sich zunächst die Warteschlange des betroffenen Ausgangs. Hält die Überlastung

[2]Reale Router, vor allem Hochleistungsrouter, sind deutlich komplexer aufgebaut als in diesem einfachen Modell dargestellt. Sie besitzen auch Eingangs-Puffer und bestehen oft aus mehreren Prozessoren, die auf der Basis von Distributed-Memory-Architekturen Multi-Processing betreiben und über interne Hochgeschwindigkeitsbusse verbunden sind. Für eine allgemeine Betrachtung, wie sie hier benötigt wird, reicht aber das vorgestellte, einfache Modell, da es die wesentlichen Eigenschaften gut beschreibt.

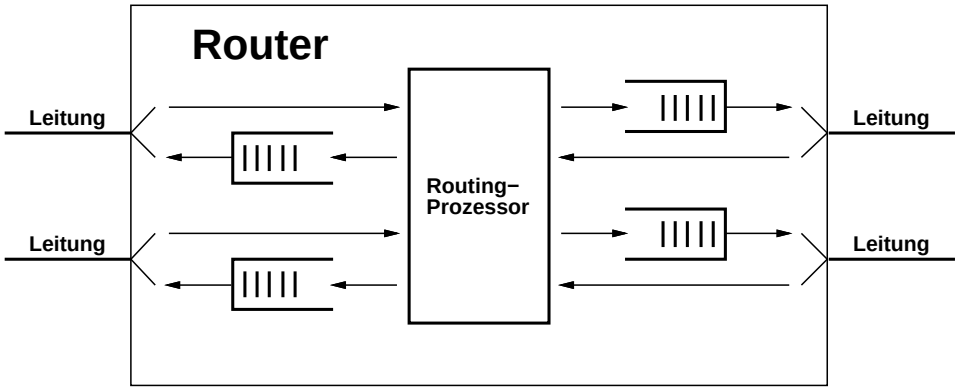

Abbildung 1.2: Warteschlangen in einem Internet-Router.

des Ausgangs an, so ist nach einer gewissen Zeit der Pufferspeicher erschöpft und es müssen Pakete verworfen werden.

Das Puffern geschieht in der Regel nach dem First-In-First-Out-Prinzip (FIFO), bei dem die Pakete die Warteschlange in der gleichen Reihenfolge verlassen, wie sie sie betreten haben. Alternativen zu diesem Prinzip werden seit einiger Zeit diskutiert und sollen bei gleicher Infrastruktur einzelnen Diensten eine bessere Dienstgüte bieten oder auch die Fairness bei der Aufteilung der verfügbaren Bandbreite auf mehrere Anwendungen verbessern [34, 81]. Diese Verfahren sind allerdings noch nicht für die Internet-Kernnetze mit ihrem stark aggregierten Verkehr geeignet und werden derzeit nur in einzelnen Teilbereichen eingesetzt. Generell bleibt festzuhalten, dass eine starke Auslastung einer Leitung durch die notwendige Pufferung zu längeren Paketlaufzeiten führt.

Das Verwerfen der Pakete wird klassischerweise nach dem **Tail-Drop**-Prinzip durchgeführt: wenn der verfügbare Pufferspeicher voll ist, werden neu hinzukommende Pakete verworfen. Auch hier kommen bereits Alternativen zum Einsatz [23], die das Verhalten der Transport- und Anwendungsschicht in Überlastsituationen optimieren sollen. Unabhängig von der speziellen Taktik ist das Verwerfen von Paketen in Routern aufgrund temporär (oder dauerhaft) überlasteter Leitungen für den überwiegenden Teil aller **Paketverluste** im Internet verantwortlich. Alle anderen Ursachen für mögliche Paketverluste, beispielsweise Bitfehler bei der Übertragung, kommen um Größenordnungen seltener vor.

Die *Paketlaufzeit eines Routers* τ_R ergibt sich im wesentlichen aus der Bearbeitungszeit τ_B, die der Routing-Prozessor zum Bearbeiten eines Pakets braucht, und der Wartezeit τ_W, die ein Paket in der Warteschlange verbringt:

$$\tau_R = \tau_B + \tau_W \tag{1.5}$$

Während τ_R in typischen Routern praktisch konstant und kleiner als eine

Millisekunde ist, schwankt τ_W sehr stark in Abhängigkeit von der momentanen Auslastung der Ausgangsleitungen und kann durchaus im Bereich von einigen zehn oder hundert Millisekunden liegen.

Verbindungen Von seinem Prinzip her ist das Internet *verbindungslos* — jedes Paket wird unabhängig von anderen Paketen und allein aufgrund der in ihm selbst enthaltenen Informationen weitergeleitet. Damit unterscheidet es sich wesentlich von *verbindungsorientierten* Netzen wie dem Telefonnetz oder auf ATM [27, 29] basierenden Netzen, in denen das Netz selbst für jede Anwendung eine **Verbindung** von einem Endpunkt zu einem anderen Endpunkt einrichtet und unterhält. Auch für Internet-Anwendungen gibt es das Konzept von Verbindungen, allerdings werden diese allein von der Transportschicht der jeweiligen Endpunkte eingerichtet und unterhalten.

Pfad Ein **Pfad** ist im Netz-Modell eine Menge zusammenhängender Knoten und Kanten, die den genauen Weg bezeichnet, den ein Paket vom Sender zum Empfänger durch das Netz nimmt. Im Internet muss nicht jedes Paket mit gleichem Sender und Empfänger den gleichen Pfad nehmen: Zum einen können Pfade im Laufe der Zeit wechseln und zum anderen kann es auch zum selben Zeitpunkt mehrere gleichwertige Pfade zwischen einem Sender und Empfänger geben. In den meisten Fällen geht man aber davon aus, dass ein Pfad oder zumindest der größte Teil eines Pfads sich über einen längeren Zeitraum hinweg nicht ändert.

Die *Paketlaufzeit eines Pfades* τ_P ergibt sich aus der Summe der Paketlaufzeiten der Leitungen und Router des Pfades, also

$$\tau_P = \sum_{i=1}^{n} \tau_{L,i} + \sum_{i=1}^{n-1} \tau_{R,i} \tag{1.6}$$

$$= \sum_{i=1}^{n} (L/B_i + l_i/c_i) + \sum_{i=1}^{n-1} (\tau_{B,i} + \tau_{W,i}) \tag{1.7}$$

wenn der Pfad zwischen Sender und Empfänger aus n Leitungen und $(n-1)$ Routern besteht; $\tau_{L,i}$, B_i, l_i und c_i die Paketlaufzeit, Bandbreite, Länge und Signalausbreitungsgeschwindigkeit der i-ten Leitung sind und $\tau_{R,i}$, $\tau_{B,i}$ und $\tau_{W,i}$ Paketlaufzeit, Bearbeitungszeit und Wartezeit des i-ten Routers sind. Dabei sind Sendezeit, Ausbreitungszeit und Bearbeitungszeit weitgehend konstante Summanden, die rein der Netztopologie abhängen, während die Wartezeit dynamisch ist und von der Verkehrslast abhängt. Auch die Paketverluste sind dynamisch und hängen von der Verkehrslast ab.

Für die Charakterisierung eines Pfades lässt sich auch seine *Bandbreite* bestimmen. Sie entspricht der kleinsten Bandbreite aller Leitungen des Pfades. In der Praxis haben die Kernnetze des Internet meist durchweg große Bandbreiten und führen stark aggregierten Verkehr, so dass selten die

Bandbreite, sondern meist Paketlaufzeit und Paketverluste die interessanten Größen eines Pfades sind.

Provider Ein **Provider** ist eine Instanz, die einen Dienst erbringt. Im Zusammenhang mit dem Internet wird „Provider“ oft als Abkürzung von **Internet Service Provider** (**ISP**) verwendet, so auch in der vorliegenden Arbeit. Ein ISP ist ein Anbieter, typischerweise eine regional oder überregional arbeitende Firma, der Endkunden Zugang zum Internet bietet. Dazu gehört neben viele möglichen Zusatzleistungen in der Regel eine Zugangsleitung, vor allem aber auch ein ↑Transitabkommen.

Internet-Zugangspunkt Ein **Internet-Zugangspunkt** ist ein Punkt, an dem ein Internet Service Provider einem Kunden Zugang zum Internet gewährt. Im hier verwendeten Netzmodell kann das jeder Knoten sein.

Struktur des Internet Will man das Internet mit einem einzigen Satz beschreiben, so sieht dieser meist wie folgt aus:

> „Das Internet ist das Netz der Netze“

Dieser Satz gilt auf mehr als nur einer Ebene. Beispielsweise kann eine Organisation oder Firma als Internet-Endkunde ein eigenes IP-Netz betreiben, das aus mehreren Ethernet-Netzen besteht. Ihr Provider betreibt in der Regel wiederum ein eigenes Netz, über das er seine Konnektivität an mehrere Kunden verteilt. Und er wendet sich selber an einen anderen, übergeordneten Provider, einen **Provider-Provider**, der ein eigenes Backbone-Netz betreibt und mehrere kleinere Provider versorgt. Das kann unter Umständen noch über weitere Stufen laufen bis hin zu großen Netzbetreibern mit kontinentalen oder globalen Backbone-Netzen.

Auf allen Ebenen kann es sowohl Kunden-Provider- als auch Provider-Provider-Beziehungen geben. Dabei verwendet man unter anderem sogenannte **Transitabkommen**, bei denen ein (kleinerer) Netzbetreiber bei einem anderen (größeren) Netzbetreiber die Verbindung zu allen weltweiten Netzen einkauft. **Peering-Abkommen** werden vornehmlich zwischen etwa gleich großen Netzbetreibern abgeschlossen, die sich über eine gemeinsam betriebene Verbindungsleitung nur gegenseitigen Zugriff auf das eigene Netz bieten, aber darüber keine Daten des einen Partners an andere Partner weiterleiten. Schließlich gibt es **Exchange-Points** („GIX“, „CIX“, etc): Orte, an denen viele Netzbetreiber mit einer Außenstelle präsent sind und alle miteinander Peering- oder Transit-Abkommen treffen.

1.3 Vorüberlegungen

Im Folgenden werden einige Überlegungen vorgestellt, die der vorliegenden Arbeit vorangegangen waren und die zur gewählten Vorgehensweise geführt haben.

1.3.1 Wahl der zu betrachtenden Schicht

Netzbetreiber und Internet-Service-Provider bieten ihre Dienste zunächst einmal auf der *Netzschicht* an. Sie transportieren für ihre Kunden IP-Pakete von und zu anderen Benutzern oder leiten sie an andere Netzbetreiber weiter. Weitere Dienste wie etwa DNS-, WWW-Cache-, NetNews- und Mail-Dienste [10, 32, 42, 68] gehören zwar typischerweise auch zum Angebot eines Providers, sind aber heute keineswegs selbstverständlich und fehlen teilweise bei Call-by-Call-Angeboten für den Internetzugang. Es handelt sich dabei auf jeden Fall um gesondert zu betrachtende Zusatzleistungen.

Den Benutzer interessiert letztlich die Performance der genutzten *Anwendungen*. Diese ergibt sich aus dem Zusammenwirken aller Komponenten aller beteiligten Schichten, schließt also nicht zuletzt auch die Performance des jeweiligen Servers mit ein. Dieser kann in einem lokalen Netz des Providers stehen, oder auch irgendwo im Internet. Je nach Last, Art und Lage des benutzten Servers kann die Performance der Anwendung beschränkt werden durch:

1. die Performance des Server,
2. die Leistungsfähigkeit (Bandbreite) der Zugangsleitung zum ISP oder
3. die Qualität der Netzverbindung zwischen ISP und den weltweit verteilten Servern.

Für jeden dieser Punkte hat der Provider die Möglichkeit, durch geschickte Planung und Konfiguration und/oder durch zusätzliche Investitionen seine Leistungsfähigkeit zu verbessern.

Seine *Server* kann ein Provider leistungsstärker machen, indem er in bessere oder zusätzliche Hardware investiert, oder möglicherweise indem er leistungsfähigere Software einsetzt.

Er kann seinen Kunden einen besseren *Zugang* ermöglichen, indem er mehr Einwahlmöglichkeiten zu Verfügung stellt oder leistungsfähigere Techniken anbietet (ISDN, xDSL, Kabelmodems, Standleitungen unterschiedlicher Übertragungskapazitäten etc.).

Seine weltweite *Internet-Anbindung* kann er verbessern, indem er leistungsfähigere Standleitungen zu seinem Provider-Provider anmietet, weitere Transitabkommen mit Netzbetreibern abschließt oder zusätzliche Peering-Verbindungen zu anderen Netzbetreibern einrichtet.

Die Performance der ersten beiden Punkte ist teilweise trivial und in anderen Fällen bereits Gegenstand laufender Untersuchungen, beispielsweise bei der Bewertung von HTTP-Servern [8, 83]. Der Gegenstand dieser Arbeit soll daher der dritte Punkt, die *Leistungsfähigkeit der Netzschicht* sein.

1.3.2 Typische Anwendungen

Bei der Vielzahl der Anwendungen im Internet und der Möglichkeiten ihres speziellen Einsatzes ist es unrealistisch, alle einzeln zu untersuchen. Es ist jedoch möglich, einige *Klassen* typischer Anwendungen zu beschreiben und zu untersuchen, in die sich meisten Internet-Anwendungen mit gute Näherung einordnen lassen:

Dateiübertragung Hier werden vergleichsweise große Datenmengen in einem Stück mittels TCP übertragen. Charakteristisch ist, dass der Aufwand für den Verbindungsaufbau sehr klein ist im Vergleich zum Aufwand für die eigentliche Datenübertragung und dass das Einschwingverhalten der TCP-Regelungsmechanismen vernachlässigt werden kann. Hierfür sind kurze Paketlaufzeit und möglichst niedrige Paketverluste wichtig. Typische Beispiele sind die Übertragung größerer Dateien mittels FTP oder HTTP oder das Verschieben großer Datenmengen mittels `ttcp`.

Einzelverbindungs-Transaktion Auch hier kommt TCP als Transportprotokoll zum Einsatz. Allerdings werden nur kleine bis mittelgroße Dateien am Stück übertragen und es wird für jede Übertragung eine neue TCP-Verbindung aufgebaut. Der Aufwand für den Verbindungsaufbau kann dabei genauso wenig vernachlässigt werden wie die TCP-Regelungsmechanismen für neue Verbindungen. Auch hier sind Paketlaufzeit und Paketverlustrate die entscheidenden Netzgrößen, wirken sich aber etwas anders aus. Das wichtigste Beispiel hierfür ist der Abruf kurzer WWW-Seiten mittels HTTP. Aber auch das Versenden kurzer E-Mail über SMTP verhält sich ähnlich.

Echtzeitanwendung Hier kommt nicht TCP zum Einsatz, sondern beispielsweise UDP mit RTP. Die Anwendung hat dabei besondere Anforderungen an die Einhaltung einer bestimmten Höchstlaufzeit aller Pakete. Dafür können Paketverluste im begrenzten Maße gegebenenfalls toleriert werden. Beispiele sind Online-Konferenzen mit Audio und/oder Video-Übertragung.

In Kapitel 7 wird näher untersucht, welchen Anteil am Internet-Verkehr diese Anwendungsklassen tatsächlich haben.

1.3.3 Einzelmessungen

Im Internet existiert bereits eine Reihe von Werkzeugen zur Messung verschiedener Leistungsmerkmale von IP-Netzen. Beispielsweise `traceroute` [38] zum Erkunden des konkreten Pfads von IP-Paketen auf dem Weg von einem Host im Internet zu einem anderen, `ttcp` [56], `netperf` [18] und `treno` [45] zum Ermitteln des möglichen Durchsatzes und `ping`[3] zum Testen der Erreichbarkeit eines entfernten Hosts im Internet. Aktuelle Arbeiten [17, 54] geben eine umfassende Übersicht über aktuelle Methoden und Werkzeuge. Nichtsdestoweniger bringen intensive Untersuchungen der tatsächlichen Performance im Internet immer wieder neue Erkenntnisse über überraschende Phänomene [61, 62, 73].

Viele der bekannten Messmethoden wie zum Beispiel `ttcp`, `netperf`, und `treno` ermitteln den möglichen Durchsatz einer Verbindung, indem sie einfach selber so viele Daten übertragen wie möglich. Diese Strategie hat erhebliche Nachteile [59]: Die von diesen Messungen erzeugte Netzlast verändert aktiv die Situation im Netz und kann so unter Umständen selbst erst die Überlast erzeugen, die dann zu ungewöhnlichen Performanceverlusten führt. Außerdem stellt diese Netzlast einen Kostenfaktor dar, der häufige und regelmäßige Messungen vieler Einzelverbindungen praktisch verbietet.

Andere Strategien wie etwa die von `ping` kommen mit sehr wenig aktiver Netzlast aus und sind so für umfangreichere Messungen viel eher geeignet. Dafür sind sie auch weniger genau: sowohl in bezug auf die Stichprobengröße einer statistischen Auswertung, als auch in der Frage, welche Bedeutung ihre Ergebnisse quantitativ für die Performance der Anwendungsschicht hat.

1.3.4 Objektivität

In den meisten Untersuchungen, die eine Bewertung der generellen Internet-Performance zum Ziel haben, wird gezielt eine kleine Zahl Server ausgewählt, die weltweit verteilt sind und dann das ganze Internet repräsentieren sollen. Das ist zum Beispiel eine Reihe von FTP-Servern [89] oder eine Reihe von WWW-Servern [52, 53].

Dieses Vorgehen ist bei Benchmarks in der Anwendungsschicht (FTP beziehungsweise HTTP in den oben genannten Untersuchungen) praktisch die einzige Möglichkeit und liefert für eine erste Näherung gute Ergebnisse. Um aber ein allgemeingültiges Maß für Internet-Performance zu definieren, muss dieses *objektiv*, also frei von einer willkürlichen Auswahl sein. Ansonsten wäre es ein leichtes, das Gesamtergebnis einer Messung allein durch eine geschickte Auswahl der Messstrecken zu manipulieren.

Schmalbandige Messungen nach Kapitel 1.3.3 bieten die Möglichkeit, eine große Anzahl verschiedener Verbindungen zu testen. Selbstverständlich

[3] `ping` ist ein Standardprogramm, das praktisch jeder TCP/IP-Implementierung beiliegt. Eine nähere Beschreibung findet sich in Kapitel 5.

wird es auch damit nicht möglich sein, das ganze Internet auszumessen. Wenn man aber nur die Internet-Hosts betrachtet, die für eine gegebene Benutzergruppe die größte Bedeutung haben und mit denen die Benutzergruppe den überwiegenden Teil ihre gesamten Internet-Verkehrs abwickelt, dann kann es ausreichen, die Verbindungen zu einer großen aber realistischen Anzahl Hosts zu untersuchen. Das Ergebnis ist eine Stichprobe, die speziell für den Bedarf einer gegebenen Benutzergruppe mit guter Näherung das ganze Internet repräsentiert.

Eine *objektive* Methode zur Bestimmung einer repräsentativen Liste von Internet-Hosts besteht also etwa darin, den tatsächlichen Internet-Verkehr einen bestimmten Benutzergruppe eine Zeit lang zu beobachten und aus den Beobachtungen eine Liste der Internet-Hosts zu ermitteln, mit denen die Benutzergruppe den meisten Verkehr austauscht.

1.3.5 Praktische Anwendbarkeit

Ein wichtiges Merkmal für die praktische Anwendbarkeit von Messmethoden ist die in 1.3.3 geforderte Schmalbandigkeit, damit die Kosten in Grenzen gehalten werden können.

Eine Reihe von Messwerkzeugen haben bestimmte Anforderungen an das Netz beziehungsweise an die Gegenstelle. So können beispielsweise `ttcp` und `netperf` nur eingesetzt werden, wenn gleichzeitig am anderen Ende der Verbindung das gleiche Programm gestartet wird (`ttcp`) oder zumindest ein passender Service installiert wurde (`netperf`). Für einen breiten Einsatz im ganzen Internet sind jedoch nur Methoden wie `treno` und `ping` geeignet, die auf normalem Verhalten von Routern oder Standard-Diensten beruhen. Der Verzicht auf eine spezielle Kooperation der Gegenstelle schränkt leider die Möglichkeiten der Messungen ein und bringt häufig unerwünschte und unberechenbare Nebenwirkungen mit sich. So kommt es durchaus vor, dass ICMP-Pakete im Netz mit anderer Priorität weitergeleitet werden als TCP- oder UDP-Pakete [21, 73].

Hier kommt es also darauf an, Messmethoden zu finden, die zum einem nur auf weit verbreiteten Features beruhen und bei denen sich zum anderen der Messverkehr möglichst wenig von normalem Verkehr unterscheidet.

1.4 Lösungsansatz

Aufgrund der genannten Überlegungen wurde ein Lösungsansatz gewählt, der aus folgenden drei Teilaufgaben besteht:

1. Es sollen Methoden entwickelt werden, mit denen aufgrund von Beobachtungen des Internet-Verkehrs gegebener Benutzergruppen automatisiert geeignete Listen der wichtigsten Internet-Hosts erstellt werden können.

2. Es sollen Messverfahren entwickelt und auf ihre Brauchbarkeit und Genauigkeit hin untersucht werden, die Messungen der grundlegenden Performance-Größen der Netzschicht auf eine Art und Weise ermöglichen, die auch bei Messungen zu vielen, weltweit verteilten Internet-Hosts hin effizient ist und keine spezielle Kooperation der Gegenseite erfordert.

3. Es soll untersucht werden, welche Bedeutung die gemessenen Größen der Netzschicht für die Performance typischer Klassen von Internet-Anwendungen haben; wie sich also anhand der Netzschicht-Performance konkrete, für den Endbenutzer verständliche Aussagen über die Performance der Anwendungsschicht machen lassen.

In Kapitel 2 wird zunächst untersucht, welche anderen Arbeiten es weltweit gibt, die eine ähnliche Zielsetzung wie die vorliegende Arbeit haben, und welche Ergebnisse sich daraus gegebenenfalls verwenden lassen. Kapitel 3 dient zur Definition der zu messenden Größen der Netzschicht. In Kapitel 4 werden Möglichkeiten zur Beobachtung des Internet-Verkehrs einer gegebenen Benutzergruppe erörtert und es wird eine Implementierung zur Erstellung einer Liste der am meisten verwendeten Internet-Server vorgestellt. Außerdem wird auf Probleme eingegangen, die sich bei groß angelegten Messungen in der Praxis ergeben haben, und es wird dargestellt, wie sie mit geeigneten Maßnahmen im Rahmen der Hostlisten-Erstellung gelöst oder zumindest gemildert werden können. In den Kapiteln 5 und 6 werden die im Rahmen der Arbeit implementierten Messverfahren vorgestellt und untersucht. Es wurden zwei verschiedene, mit unterschiedlichen Pakettypen arbeitenden Verfahren entwickelt und jedes wurde schrittweise aufgrund gesammelter Erfahrungen verbessert. Kapitel 7 schließlich behandelt die Frage, wie sich mit den Messergebnissen Aussagen über die Performance der Anwendungsschicht machen lassen. Dazu werden im wesentlichen Modelle aus der jüngeren Literatur herangezogen. Zur Validierung des gesamten Ansatzes wird in Kapitel 8 die aufgrund der Messungen nach Kapitel 6 und der Modelle nach Kapitel 7 geschätzte Performance der Anwendungsschicht mit real gemessener Anwendungsperformance verglichen. Kapitel 9 fasst schließlich die Arbeit noch einmal zusammen und nennt mögliche Felder für weitere Forschungsaktivitäten.

Kapitel 2

Andere Arbeiten

Es gibt weltweit einige andere Arbeiten, die sich mit ähnlichen Zielen beschäftigen wie die vorliegende Arbeit:

2.1 Network Performance Index

Vor einigen Jahren wurde ein **Network Performance Index** für das European Academic and Research Network (EARN) entwickelt [12, 13]. Analog zu dem in der Finanzwelt üblichen „Index“ für Börsenkurse werden dabei verschiedene Leistungswerte mehrerer Verbindungen in einem Wert zusammengefasst, um den Zustand und die Leistungsfähigkeit des Netzes kurz und prägnant zu dokumentieren. Dieser „Network Performance Index“ wurde regelmäßig erstellt und über längere Zeiträume protokolliert.

Für das Internet fehlt ein derartiger Index. Allerdings ist das Internet im Vergleich zum EARN viel umfangreicher, komplexer und dynamischer. Es ist topologisch nie vollständig zu erfassen, geschweige denn auszumessen. Außerdem fehlt bei der Heterogenität der Infrastruktur und der Vielfalt der Anwendungen jegliche Vorstellung davon, was ein solcher Index für das Internet überhaupt aussagen sollte.

2.2 Internet Weather Report

Die Matrix Information and Directory Services Inc. (MIDS) in Texas, USA, gibt einen regelmäßigen „Wetterbericht“ [69] heraus, der die aktuelle Gesamtsituation des Internets darstellt. Dazu werden in regelmäßigen Abständen ICMP-Echo-Request-Pakete („`ping`-Pakete“) an viele, weltweit verteilte Rechner des Internets geschickt. Genau sechs mal pro Tag wird so die Verbindung in alle Teile der Welt überprüft. Die Ergebnisse werden in geographischen Karten dargestellt — ganz analog zu einem meteorologischen Wetterbericht. Bild 2.1 zeigt ein Beispiel einer solchen „Internet-Wetterkarte“. Jede Messung wird durch einen Kreis dargestellt, wobei der Durchmes-

ser die gemessene Paketlaufzeit wiedergibt: Je größer der Kreis, desto länger hat das `ping`-Paket gebraucht, desto schlechter ist also die Verbindung dorthin. Mehrere Messungen mit gleichen Ergebnis am gleichen geographischen Ort werden durch verschiedene Farben der Kreise dargestellt. In Analogie zu meteorologischen Schlechtwettergebieten gibt es Häufungen von großen Kreisen, die Gegenden mit besonders schlechter Internet-Konnektivität anzeigen.

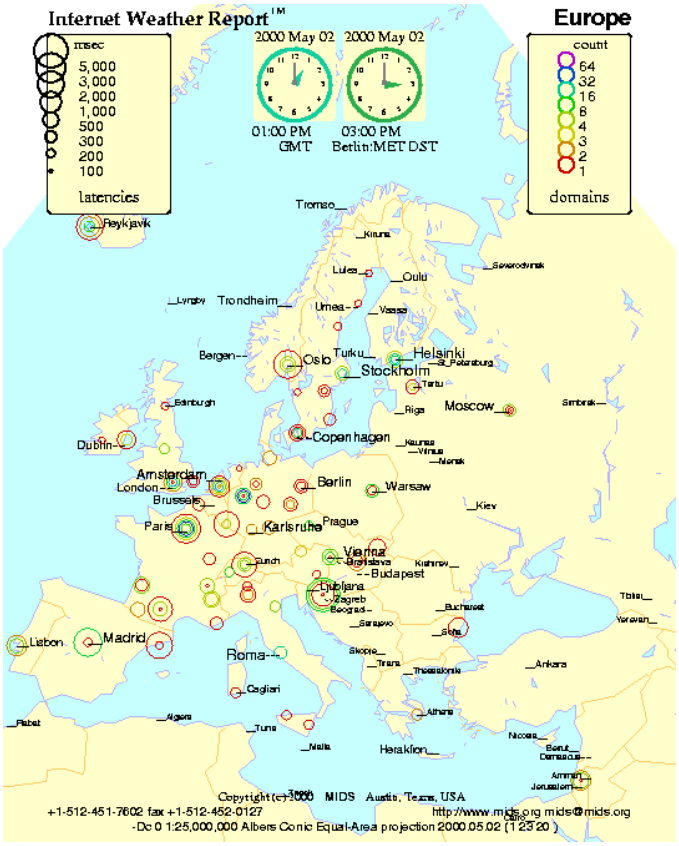

Abbildung 2.1: Internet Weather Report für Europa am 2.5.2000 um 15:00.

Der Internet Weather Report veranschaulicht auch Tagesverläufe des „Internet-Wetters“. Dazu werden die Plots verschiedener Uhrzeiten eines Tages in einer Animation zusammengestellt.

Interessant ist die Frage, wie die Rechner ausgewählt wurden, die im Internet Weather Report auf ihre Erreichbarkeit hin getestet werden. Dazu wurde eine Liste aller nennenswerten Domains des Internet herangezogen [43] und der Mail-Exchanger für jede Domain ermittelt. So wird sichergestellt, dass man eine Liste von wirklich weltweit verteilten Rechnern hat und dass aus jeder administrativen Einheit des Internet zumindest ein Rechner

getestet wird. Damit ist allerdings nicht gewährleistet, dass dieser Rechner in seiner Internet-Konnektivität wirklich typisch für die Domain ist, die er vertritt. Ein Mail-Exchanger kann topologisch ganz anders im Internet angeordnet sein als die anderen Rechner der Domain, für die er steht. In den meisten Fällen wird er aber recht „nahe“ bei Ihnen liegen.

Die geographische Darstellung ist mit Sicherheit eine sehr interessante Art der Visualisierung. Sie ist aber nicht besonders geeignet, die Unterschiede verschiedener Internet Service Provider aufzuzeigen, da sich die Betätigungsbereiche vieler Provider geographisch stark überlappen.

Außerdem macht die geographische Darstellung nicht unbedingt klar, welche Bedeutung bestimmte Ergebnisse für welchen Benutzer haben. In vielen Fällen weiß der Benutzer nämlich wenig darüber, in welcher geographischen Region die Internet-Server liegen, die er häufig benutzt. Selbst wenn, kann er aus den „Wetterkarten“ oft nicht ablesen, welche Kreise in der Region für seinen Server stehen.

Genauso wenig wird erörtert, welche Bedeutung bestimmte Paketlaufzeiten für den Benutzer haben – sie werden einfach roh und unkommentiert angegeben. Dies ist insofern gar nicht verkehrt, als eine genaue Interpretation der Bedeutung von Paketlaufzeiten ohnehin sehr schwierig, wenn nicht unmöglich ist. Davon unabhängig wird leider nicht dokumentiert, ob und, falls ja, wie *Paketverluste* in der Darstellung berücksichtigt werden.

2.3 Lachesis

Von der gleichen Motivation wie diese Arbeit getrieben, entstand in der Firma Intel das Projekt **Lachesis** [75], das es sich zur Aufgabe gemacht hat, einen Benchmark für Internet Service Provider zu entwickeln. Er soll große Internet-Kunden – wie Intel – in die Lage versetzen, die Qualität verschiedener ISP direkt miteinander vergleichen zu können.

Anders als viele andere Untersuchungen konzentriert sich Lachesis darauf, Paketlaufzeiten (auch RTT, Round Trip Time) und Paketverluste zu untersuchen. Durchsatzzahlen sind hingegen kein primäres Ziel, da sie nur bedingt und nur in bestimmten Fällen ein vorrangiges Gütekriterium für die verschiedenen Internet-Anwendungen sind.

Ein weiteres interessantes Konzept von Lachesis ist das der „Landmarks“. Bei Lachesis' Landmarks handelt es sich um eine Liste von Rechnern im Internet, zu denen die Konnektivität gemessen wird. Dabei gibt es sowohl eine Default-Liste als auch die Möglichkeit, diese Liste an den Bedarf des jeweiligen Benutzers anzupassen. Die Default-Liste besteht aus wichtigen Servern: den root-Nameservern sowie häufig benutzten FTP- und WWW-Servern.

Die Messungen können sowohl am POP des zu messenden Service Providers durchgeführt werden als auch auf der Seite des Kunden. Zusammen

mit der Anpassbarkeit der Landmark-Liste erlaubt das den Benutzern, jederzeit nach Belieben eigene Benchmarks durchzuführen und die Ergebnisse mit ihrem Bedarf zu parametrisieren. Leider wird kein Hilfsmittel mitgeliefert und keine Methode vorgeschlagen, wie der Benutzer seinen „Bedarf“ genauer ermitteln und geeignet formulieren könnte.

Zur Ermittlung von Paketlaufzeit und Paketverlust benutzt Lachesis ein Programm namens FPING, das übliche ICMP-Echo-Requests verschickt. Genau wie für den Internet Weather Report bedeutet das, dass die Messergebnisse sehr leicht verfälscht werden können, da große Netzbetreiber ICMP-Pakete häufig anders als andere Pakete behandeln [21].

Auch finden sich hier keine Aussagen über die eigentliche Bedeutung der Ergebnisse für die Paketlaufzeit und Paketverluste. Es fehlt an Methoden, aus beiden Werten sinnvolle Aussagen über die Performance der Anwendungsschicht abzuleiten.

2.4 PingER

Am Stanford Linear Accelerator Center (SLAC), einem staatlichen Forschungslabor der USA an der Stanford University, wird das Projekt **pingER** betrieben, das für die internationale Gemeinschaft der Hochenergie-Physik (High Energy Nuclear and Particle Physics) die vorhandene Internet-Infrastruktur untersucht [49]. Seit 1995 werden in diesem Rahmen laufend die Internet-Verbindungen zwischen den beteiligten Universitäten und Instituten untersucht. Dazu werden auch hier mit Hilfe von `ping` Paketlaufzeit und Paketverluste gemessen.

Im Rahmen dieses Projekts wurden bereits schlechte Erfahrungen mit dem Einsatz von ICMP als Pakettyp gemacht. Als bestimmte Netzbetreiber als Maßnahme gegen Denial-of-Service-Angriffe eine Drosselung dieses Pakettyps einführten, stiegen die von pingER gemessenen Paketverluste zu bestimmten Einrichtungen stark an, ohne dass die Netzverbindung wirklich schlechter wurde. Auf dieses Problem wurde und wird mit zwei Maßnahmen reagiert:

- Man versucht, von solchen Drosselungs-Maßnahmen betroffene Internet-Pfade von Hand ausfindig zu machen und sie von allgemeinen Bewertungen der Netzqualität auszunehmen.
- Für die Zukunft plant man, die Messungen nicht mehr oder nicht nur mit ICMP-Paketen, sondern auch mit TCP-Syn-Paketen durchzuführen. Dies wurde aber bislang noch nicht umgesetzt.

Der grundlegende Unterschied zwischen dem Projekt pingER und der vorliegenden Arbeit ist, dass sich pingER auf eine feste, vorgegebene Benutzergruppe bezieht und nur zwischen bestimmten Einrichtungen Messungen

durchführt, während die vorliegende Arbeit auf Methoden zielt, die sich jederzeit auf beliebige Benutzergruppen anwenden lassen. Des weiteren ist die vorliegende Arbeit fortschrittlicher in Bezug auf die Messtechnik, da Probleme mit ICMP von Anfang an vermieden wurden und da die IETF-Empfehlungen für aktive Messungen durchgehend befolgt werden, was bei pingER nicht der Fall ist.

Ein besonders schöner Aspekt von pingER ist, dass mittlerweile für viele Jahre Ergebnisse von gleich bleibenden Messungen vorliegen, so dass die langfristige Entwicklung (und Verbesserung) der Qualität der weltweiten Internet-Verbindungen gut dargestellt werden kann.

2.5 IETF-Arbeitsgruppe IP Performance Metrics

In Rahmen der Internet Engineering Task Force (IETF) [35] gibt es eine Arbeitsgruppe zum Thema „Internet Performance and IP Provider Metrics“ (**IPPM**) [36]. Sie war früher der „Benchmarking Methodology Working Group“ (bmwg) [11] untergeordnet, ist aber seit November 1997 eine eigenständige Arbeitsgruppe. Dieser Wechsel hat sich als sinnvoll erwiesen, weil die Zusammenhänge zwischen Performance-Maßen der Netz-Schicht (IP) einerseits und Protokollen der darüberliegenden Transport-Schicht (TCP) andererseits immer größere Beachtung finden und weil sich mit den übrigen Arbeiten der Gruppe BMWG, die sich mehr mit der Verbindungsschicht beschäftigt, vergleichsweise wenig Gemeinsamkeiten finden ließen. Neben der offiziellen WWW-Seite der Arbeitsgruppe bieten einige beteiligte Wissenschaftler weitere Informationen zum Thema [1, 44].

Die IPPM-Gruppe hat es sich zur Aufgabe gemacht, Werkzeuge zur Ermittlung genauer, quantitativer Leistungs-Daten von Internet-Pfaden zu liefern. Dafür sollen sowohl gängige Begriffe und Größen, sowie ihre Beziehung zu bekannten Messmethoden genau definiert werden und es sollen neue Größen und Messmethoden entwickelt werden. Die Ergebnisse sind bereits zum großen Teil als RFC-Dokumente veröffentlicht worden. So wird in RFC 2330 [59] ein Modell des Internet beschrieben, es werden Anforderungen an sinnvolle Größen definiert und es werden verschiedene Arten von Messmethoden unterschieden. Des weiteren gibt es Dokumente über Konnektivität (connectivity), Paketverluste (packet loss) und Paketlaufzeiten (packet delay).

Die Dokumente der IPPM-Gruppe bieten eine gute Grundlage für die in dieser Arbeit verwendeten Größen. Teilweise kann auf von der IPPM-Gruppe definierte Größen zurückgegriffen werden, teilweise müssen aber auch neue oder erweiterte Größen eingeführt werden.

2.6 Implementierungen von IPPM-Maßen

Für jedes von der IETF verabschiedete Protokoll gilt, dass es anfangs den Status eines zeitlich begrenzten „Proposed Standard“ hat. Werden zwei unabhängige Implementierungen vorgelegt und kann die reibungslose Zusammenarbeit dieser Implementierungen gezeigt werden, kann es zum immer noch zeitlich begrenzten „Draft Standard“ erhoben werden. Erst danach kann es zum eigentlichen „Standard“ werden und damit eine unbegrenzte Lebensdauer erhalten. Für die IPPM-Maße strebt man ein analoges Verfahren an, bei dem zwei unabhängige Implementierungen von Messmethoden bei gleichen Parametern die gleichen Ergebnisse liefern sollen. So trifft es sich gut, dass sich tatsächlich zwei unabhängige Projekte mit der Implementierung solcher Messmethoden beschäftigen:

Surveyor Die Advanced Network & Services Inc. und die Common Solutions Group, eine Gruppe amerikanischer Universitäten, betreiben das Projekt Surveyor, das Messmethoden für die in der IPPM-Gruppe definierten Größen unidirektionaler Paketlaufzeit und -Verluste implementiert und sie fortwährend zwischen den Standorten der Teilnehmer misst [80].

RIPE Test Traffic Project Die Organisation „Réseaux IP Européens“ (RIPE) [71] betreibt das „Test Traffic Project“ [85].

Beide Projekte haben jeweils ein Modell einer Messstation oder „Testbox“ entwickelt — ein Host, der eine über einen GPS-Empfänger genau synchronisierte Uhr besitzt, Messungen durchführt und die Ergebnisse für eine zentrale Sammelstation zu Verfügung stellt. Jedes Projekt betreibt eine Reihe solcher Messstationen, die weltweit bei kooperierenden Partnern aufgestellt sind und regelmäßig Testpakete zu den anderen Messstationen des gleichen Projekts senden. Die Messergebnisse werden jeweils zentral gesammelt und ausgewertet. Ein Vergleich zwischen diesen beiden unabhängigen Implementierungen von Messverfahren wurde mittlerweile mit positivem Ergebnis durchgeführt [84].

Beide Projekte haben eine ähnliche Intention wie die vorliegende Arbeit und zielen auf die gleichen Größen, unterscheiden sich jedoch in einem für die Praxis sehr relevantem Punkt: Sie setzen voraus, dass an beiden Enden eines zu untersuchenden Pfads kooperierende Messstationen existieren. Die hier vorliegende Arbeit setzt es sich jedoch zum Ziel, von einem Punkt aus zu anderen Teilen des Internet die Verbindungsqualität untersuchen zu können, ohne eine spezielle Kooperation der Gegenstelle zu fordern.

2.7 Application Response Measurement

Als ein Beispiel eines grundlegend anderen Ansatzes für Performance-Messungen im Internet sei hier der von der **Computer Measurement Group** (CMG) definierte Standard **Application Response Measurement** erwähnt [28]. Hier interessieren sich Anbieter von Internet-Diensten dafür, welche Internet-Performance ihre potenziellen Kunden wahrnehmen. Erreicht werden soll dies, indem die Anbieter auf ihren WWW-Seiten bestimmte aktive Inhalte einbinden, die dann von jedem Client-Rechner aus die Zeitdauer bestimmter Vorgänge der Anwendungsprotokolle bestimmen und dem Server mitteilen.

Abgesehen davon, dass hier die Initiative vom *Anbieter* von Internet-Diensten ausgeht und nicht vom *Benutzer*, liegt der grundsätzliche Unterschied dieses Projekts zur vorliegenden Arbeit darin, dass hier rein in der Anwendungsschicht statt in der Netzschicht gemessen wird. Dass dabei nicht nur die Performance des Netzes sondern auch des jeweiligen Servers und der Anwendung berücksichtigt wird, ist in diesem Fall auch durchaus erwünscht. Auch benötigt dieser Ansatz eine Kooperation der Gegenseite: ein Benutzer muss dem Installieren eines Java-Applets oder eines Active-X-Agenten in seinem WWW-Browser zustimmen.

Kapitel 3

Performance-Maße

Um die im Rahmen der vorliegenden Arbeit durchzuführenden Messungen sinnvoll planen und auswerten zu können, müssen zunächst einmal die zu betrachtenden Größen festgelegt werden. Die IPPM-Gruppe hat mehrere Anforderungen zusammengestellt, denen ihre Größen genügen sollen, zum Beispiel:

- Die Größen sollen klar definiert sein und eine konkrete Bedeutung haben.
- Eine Meßmethode für eine Größe soll wiederholbar sein, das heißt, wenn dieselbe Methode mehrmals unter denselben Bedingungen eingesetzt wird, sollen die Messungen jedes Mal dasselbe Ergebnis haben.
- Die Größen sollen Benutzern und Providern helfen, die Performance zu verstehen, die sie erhalten beziehungsweise liefern.
- Die Größen sollen den Aufbau künstlicher Performance-Ziele vermeiden.

Es werden nun durchaus Größen definiert, für die keine offensichtliche Möglichkeit zur Messung besteht. Während dies akzeptiert wird, wird aber gleichzeitig gefordert, dass eine Größe immer eine konkrete Bedeutung hat und eindeutig definiert wird. In anderen Worten: Schwierigkeiten und Ungenauigkeiten beim tatsächlichen Messen einer Größe werden hingenommen, nicht jedoch Uneindeutigkeiten bei der Definition und Bedeutung einer Größe.

3.1 Konnektivität

Konnektivität ist die Grundlage des Internet überhaupt. Eine Größe, die aussagt, ob zwei Hosts im Internet sich gegenseitig überhaupt erreichen können,

ist daher der naheliegende Ausgangspunkt für eine Sammlung von Internet-Performance-Maßen. Im folgenden werden mehrere Größen definiert, von denen einige hauptsächlich zur Definition anderer verwendet werden. Sie entsprechen den in RFC 2678 [60] definierten Konnektivitäts-Maßen.

Unmittelbare unidirektionale Konnektivität

Name: Unmittelbare unidirektionale Typ-P-Konnektivität („Type-P-Instantaneous-Unidirectional-Connectivity“).

Parameter:
- A_1: ein Host, dargestellt durch eine IP-Adresse
- A_2: ein Host, dargestellt durch eine IP-Adresse
- T: ein Zeitpunkt
- P: ein Pakettyp

Maßeinheit: Boolean.

Definition: A_1 hat *unmittelbare unidirektionale Typ-P-Konnektivität* zu A_2 zur Zeit T, wenn ein Paket vom Typ P, das von A_1 zum Zeitpunkt T zu A_2 abgeschickt wird, irgendwann bei A_2 ankommt.

Diskussion: Diese Größe ist nicht unmittelbar nützlich, weil sie sich auf einen einzigen Zeitpunkt bezieht und unidirektional ist. Die meisten Anwendungen benötigen Konnektivität in beiden Richtungen, so etwa jede TCP-Verbindung. Außerdem wird meistens Konnektivität über einen längeren Zeitraum benötigt. Schließlich kann es gut sein, dass man zu einem bestimmten Zeitpunkt keine Konnektivität hat, obwohl die Konnektivität kurz vor und kurz nach diesen Zeitpunkt sehr wohl besteht. Das kann etwa der Fall sein, wenn die Warteschlange eines Routers gerade überfüllt ist und der Router vorübergehend Pakete verwirft.

Ein wichtiger Punkt ist auch, dass nichts darüber ausgesagt wird, *wann* das Paket am Ziel ankommt. Rein theoretisch kann entsprechend RFC 791 ([65]) zwar wegen der Größe des TTL-Feldes ein IP-Paket maximal 255 Sekunden existieren. In der Praxis ist das TTL-Feld aber ein reiner Zähler der Verbindungen [9], wobei zum Durchlaufen der meisten Verbindungen weit weniger als eine Sekunde benötigt wird. Des weiteren bestehen Internet-Pfade in der Praxis meist aus wesentlich weniger Verbindungen (typischerweise unter 20), so dass ein Paket selten auch nur annähernd eine Lebensdauer von 255 Sekunden erreicht. Trotzdem ist es denkbar, dass ein Paket auch länger als 255 Sekunden von seiner Quelle zu seinem Ziel unterwegs sein kann. Diese Überlegungen sollten berücksichtigt werden, wenn Messungen dieser Größe unternommen werden.

Unmittelbare bidirektionale Konnektivität

Name: Unmittelbare bidirektionale Typ-P-Konnektivität („Type-P-Instantaneous-Bidirectional-Connectivity“).

Parameter:
- A_1: ein Host, dargestellt durch eine IP-Adresse
- A_2: ein Host, dargestellt durch eine IP-Adresse
- T: ein Zeitpunkt
- P: ein Pakettyp

Maßeinheit: Boolean.

Definition: A_1 und A_2 haben *unmittelbare bidirektionale Typ-P-Konnektivität* zur Zeit T, wenn A_1 unmittelbare unidirektionale Typ-P-Konnektivität zu A_2 hat und A_2 unmittelbare unidirektionale Typ-P-Konnektivität zu A_1 hat.

Diskussion: Alternativ könnte man auch definieren, dass A_1 und A_2 genau dann bidirektionale Konnektivität haben, wenn A_1 zur Zeit T unmittelbare unidirektionale Typ-P-Konnektivität zu A_2 hat und A_2 zur Zeit $T+\Delta T$ unmittelbare unidirektionale Typ-P-Konnektivität zu A_1 hat, wobei $T+\Delta T$ der Zeitpunkt ist, zu dem ein Paket von A_1 bei A_2 ankommt. Das wäre für Messungen besser geeignet, weil man eine Antwort von A_2 auf A_1 verwenden könnte, um die Konnektivität zu messen. Andererseits bräche es die Symmetrie der obigen Definition und würde bedeuten, dass die Paketlaufzeit von A_1 zu A_2 berücksichtigt werden muss.

Unidirektionale Intervall-Konnektivität

Name: Unidirektionale Typ-P-Intervall-Konnektivität („Type-P-Interval-Unidirectional-Connectivity“).

Parameter:
- A_1: ein Host, dargestellt durch eine IP-Adresse
- A_2: ein Host, dargestellt durch eine IP-Adresse
- T: ein Zeitpunkt
- ΔT: eine Zeitdauer
- P: ein Pakettyp

Maßeinheit: Boolean.

Definition: A_1 hat im Zeitintervall $[T, T+\Delta T]$ *unidirektionale Typ-P-Intervall-Konnektivität* zu A_2 genau dann, wenn es einen Zeitpunkt $T' \in [T, T+\Delta T]$ so gibt, dass A_1 zur Zeit T' unmittelbare unidirektionale Typ-P-Konnektivität zu A_2 hat.

Bidirektionale Intervall-Konnektivität

Name: Bidirektionale Typ-P-Intervall-Konnektivität („Type-P-Interval-Bidirectional-Connectivity“).

Parameter:
- A_1: ein Host, dargestellt durch eine IP-Adresse
- A_2: ein Host, dargestellt durch eine IP-Adresse
- T: ein Zeitpunkt
- ΔT: eine Zeitdauer
- P: ein Pakettyp

Maßeinheit: Boolean.

Definition: A_1 und A_2 haben im Zeitintervall $[T, T + \Delta T]$ *bidirektionale Typ-P-Intervall-Konnektivität* genau dann, wenn in diesem Zeitintervall A_1 unidirektionale Typ-P-Intervall-Konnektivität zu A_2 hat und A_2 unidirektionale Typ-P-Intervall-Konnektivität zu A_1 hat.

Diskussion: Diese Größe hat noch keinen praktischen Wert. In der Praxis wird nämlich meist verlangt, dass ein Paket von A_1 zu A_2 eine Antwort auslösen kann, die dann in einem Paket von A_2 zu A_1 transportiert werden kann. Mit der vorliegenden Definition wäre es durchaus möglich, dass A_1 und A_2 „volle Konnektivität“ haben, aber beispielsweise nur zur einer Zeit T' die so früh im Intervall $[T, T + \Delta T]$ liegt, dass keine Antworten mehr möglich sind. Aus diesem Manko rührt die Motivation für die folgende und letzte Konnektivitätsdefinition her.

Bidirektionale Normalkonnektivität

Name: Bidirektionale Typ-P1-P2-Intervall-Normalkonnektivität („Type-P1-P2-Interval-Temporal-Connectivity“).

Parameter:
- A_1: ein Host, dargestellt durch eine IP-Adresse
- A_2: ein Host, dargestellt durch eine IP-Adresse
- T: ein Zeitpunkt
- ΔT: eine Zeitdauer
- P_1: ein Pakettyp
- P_2: ein Pakettyp

Maßeinheit: Boolean.

Definition: A_1 hat im Zeitintervall $[T, T + \Delta T]$ *bidirektionale Typ-P_1-P_2-Intervall-Normalkonnektivität* zu A_2 genau dann, wenn Zeitpunkte T_1 und T_2 sowie die Zeitdauern ΔT_1 und ΔT_2 so existieren, dass gilt:

- $T_1, T_1 + \Delta T_1, T_2, T_2 + \Delta T_2 \in [T, T + \Delta T]$
- $T_1 + \Delta T_1 \leq T_2$
- Zur Zeit T_1 hat A_1 unmittelbare unidirektionale Typ-P_1-Konnektivität zu A_2.
- Zur Zeit T_2 hat A_2 unmittelbare unidirektionale Typ-P_2-Konnektivität zu A_1.
- ΔT_1 ist die Zeit, die ein Typ-P_1-Paket zur Zeit T_1 von A_1 nach A_2 braucht.
- ΔT_2 ist die Zeit, die ein Typ-P_2-Paket zur Zeit T_2 von A_2 nach A_1 braucht.

Diskussion: Dies ist nun endlich die Definition einer praktisch brauchbaren Größe: Ein Internet-Host kann eine Anfrage an einen anderen stellen und dieser kann seine Antwort erfolgreich übermitteln. Es wurden bewusst unterschiedliche Pakettypen für Hin- und Rückweg vorgesehen, weil es bei vielen Anwendungen tatsächlich vorkommen kann, dass Frage- und Antwortpakete unterschiedliche Typen haben.

3.2 Paketlaufzeit

Die Laufzeit eines Pakets durch das Internet von einem Sender zu einem Empfänger ist eine bedeutsame Größe für die Qualität der betroffenen Verbindung:

- Viele Anwendungen arbeiten schlecht oder gar nicht, wenn die Paketlaufzeit einen bestimmten Schwellwert überschreitet.
- Für viele Transportprotokolle ist die Paketlaufzeit von unmittelbarer Bedeutung für ihre Performance: je kürzer die Paketlaufzeit, desto höher der erzielbare Durchsatz.
- Das Minimum der Paketlaufzeit gibt einen Anhaltspunkt für den Anteil der Sende- und Ausbreitungszeit.
- Werte über dem Minimum ergeben sich typischerweise aus der Wartezeit in Routern und sind ein Anzeichen für eine höhere Auslastung von Ressourcen irgendwo im betroffenen Pfad.

So finden sich unter den Ergebnissen der IPPM-Gruppe gleich zwei RFC-Dokumente, die sich speziell mit der Paketlaufzeit befassen.

Unidirektionale Paketlaufzeit

RFC 2679 [3] definiert eine Größe für unidirektionale Paketlaufzeit:

Name: Unidirektionale Typ-P-Laufzeit („Type-P-One-way-Delay“).

Parameter:
A_1: ein Host, dargestellt durch eine IP-Adresse
A_2: ein Host, dargestellt durch eine IP-Adresse
T: ein Zeitpunkt
P: ein Pakettyp

Maßeinheit: Eine positive, reelle Zahl von Sekunden, oder undefiniert, dargestellt als Unendlich, ∞.

Definition: Die Zeitspanne ΔT ist die *unidirektionale Typ-P-Laufzeit von A_1 nach A_2 zum Zeitpunkt T* genau dann, wenn ein Paket vom Typ P, das von A_1 zum Zeitpunkt T abgesendet wird, zum Zeitpunkt $T + \Delta T$ bei A_2 ankommt. Dabei zählt beim Absenden genau der Zeitpunkt der Übertragung des ersten Bit und beim Empfangen genau der Zeitpunkt des Empfangs des letzten Bit des Pakets.

Die *unidirektionale Typ-P-Laufzeit von A_1 nach A_2 zum Zeitpunkt T* ist undefiniert genau dann, wenn ein Paket vom Typ P zum Zeitpunkt T von A_1 nach A_2 gesendet wird aber zu keinem Zeitpunkt dort ankommt.

Diskussion: Die Messung von unidirektionaler Paketlaufzeit hat gegenüber der bidirektionalen Paketlaufzeit einige Vorteile:

- Pfade sind im Internet häufig asymmetrisch und können je nach Richtung über vollkommen verschiedene Service Provider laufen. Die unterschiedlichen Eigenschaften beider Richtungen werden nur deutlich, wenn man sie auch getrennt untersucht.
- Selbst wenn ein Pfad symmetrisch ist, können beide Richtungen aufgrund unterschiedlicher Auslastung eine unterschiedliche Dienstgüte liefern.
- Auch im Zusammenhang mit der Einführung von Mechanismen für differenzierte Dienstgüte [34] können richtungsabhängige Effekte auftreten.

Diesen Vorteilen stehen allerdings auch Nachteile gegenüber:

- Die beiden an einer Messung beteiligten Hosts müssen speziell miteinander zusammenarbeiten, sich zur richtigen Zeit auf das Senden beziehungsweise Empfangen des gewünschten Pakets vorbereiten und die

Zeitpunkte bestimmen. Nach der Messungen müssen diese Ergebnisse zusammengetragen werden, um aus den verschiedenen Zeitpunkten auf die Paketlaufzeit schließen zu können.

- Um die Paketlaufzeit mit brauchbarer Genauigkeit bestimmen zu können, müssen die Uhren der beiden Hosts hinreichend genau miteinander synchronisiert werden, und zwar durch ein anderes Medium als das untersuchte Netz. Die Dokumente der IPPM-Gruppe widmen sich ausführlich dem Problem dieser Synchronisation.

Des weiteren wird eine zusammengesetzte Größe definiert:

Name: Poisson-Sequenz unidirektionaler Typ-P-Laufzeit („Type-P-One-way-Delay-Poisson-Stream“).

Parameter:

A_1:	ein Host, dargestellt durch eine IP-Adresse
A_2:	ein Host, dargestellt durch eine IP-Adresse
T_0:	ein Zeitpunkt
T_f:	ein Zeitpunkt
λ:	eine Senderate in s^{-1}
P:	ein Pakettyp

Maßeinheit: Eine Sequenz von Paaren, wobei sich ein Paar zusammensetzt aus T, einem Zeitpunkt, und ΔT, einer positiven, reellen Zahl von Sekunden, oder undefiniert.

Die Werte von T sind in der Sequenz monoton steigend. Bei jedem Paar ist T ein Parameter und ΔT ein Ergebnis der unidirektionalen Typ-P-Laufzeit.

Definition: Es wird ein (Pseudo-)Zufalls-Prozess erzeugt, der im Intervall $[T_0, T_f]$ mit der Rate λ Poisson-verteilt Werte für T erzeugt. Für jedes T wird die unidirektionale Typ-P-Laufzeit bestimmt und bildet mit diesem T ein Paar aus der Ergebnis-Sequenz.

Diskussion: Diese Sequenz bildet einen Messreihe, mit der sich die im Internet allgegenwärtigen Schwankungen gemessener Größen erfassen lassen.

Auf dieser Sequenz aufbauend werden eine Reihe statistischer Größen beschrieben, im wesentlichen das Perzentil (siehe Kapitel 3.4) der ΔT-Werte und das inverse Perzentil, sowie die Spezialfälle Median und Minimum. Bei der Bestimmung des Perzentil werden dabei Paketverluste insofern berücksichtigt, dass verlorene Pakete mit einer unendlich langen Laufzeit gezählt werden. Daher ist es möglich, dass auch ein Perzentil undefiniert sein kann.

Bidirektionale Paketlaufzeit

Analog zur unidirektionalen Typ-P-Laufzeit aus RFC 2679 beschreibt RFC 2681 [5] eine *bidirektionale Typ-P-Laufzeit* („Type-P-Round-trip-Delay“) inklusive einer *Poisson-Sequenz bidirektionaler Typ-P-Laufzeit* etc.

Im folgenden werden nur die Unterschiede der bidirektionalen zur unidirektionalen Paketlaufzeit beschrieben und diskutiert:

- Der empfangende Host A_2 sendet sofort nach dem Empfang des ersten Pakets seinerseits ein Paket zu A_1. Das Ergebnis ist die Differenz zwischen dem Zeitpunktes des Empfangs dieses zweiten Pakets bei A_1 und des Absendens des ersten Pakets von A_1.

- Es wird zwischen der *bidirektionalen Paketlaufzeit von A_1 nach A_2* und der *bidirektionalen Paketlaufzeit von A_2 nach A_1* unterschieden. Zusätzlich wird die Formulierung *bidirektionale Paketlaufzeit zwischen A_1 und A_2* zugelassen. Sie unterscheidet die Richtung nicht und kann einen der beiden erstgenannten Begriffe meinen. Diese Mehrdeutigkeit kann bewusst in Kauf genommen werden, weil der Unterschied in der Praxis meist geringfügig ist und weil dadurch gegebenenfalls eine zusätzliche Messung eingespart werden kann.

Gegenüber der unidirektionalen Paketlaufzeit hat die bidirektionalen Paketlaufzeit unter anderem folgende Vorteile:

- Für die meisten Transportprotokolle spielt nur die bidirektionale Paketlaufzeit eine Rolle. Diese direkt zu messen ist unter Umständen genauer, als sie aus den für beide Richtungen getrennt gemessenen unidirektionalen Paketlaufzeiten abzuleiten.

- Da die Zeitnahme nur bei Host A_1 erfolgt, entfällt die aufwändige Synchronisation der Uhren beider Hosts. Es muss lediglich sichergestellt werden, dass die Uhr von Host A_1 den Parameter T genau genug einhält und dass die Uhr nicht während einer laufenden Messung neu gestellt wird, oder dass einer Verstellung der Uhr Rechnung getragen wird.

- Die einzige Anforderung an Host A_2 ist, dass ein eingehendes Paket umgehend beantwortet wird durch ein Paket, das an den Absender des ersten Pakets geschickt wird. Es gibt unter den Internet-Protokollen eine Reihe Standard-Anwendungen und -Funktionen, die diese Anforderung gegebenenfalls erfüllen, ohne dass dort spezielle Software installiert werden müsste, ja sogar ohne dass der Verwalter dieses Hosts überhaupt etwas von den Messungen wissen müsste.

Vor allem aus den beiden letztgenannten Gründen wird im Rahmen der vorliegenden Arbeit allein die bidirektionale Paketlaufzeit untersucht. Untersuchungen der unidirektionalen Paketlaufzeit, die mit dem Aufbau eines

recht umfangreichen Instrumentariums auf vielen Hosts verbunden sind, werden derzeit von anderen Gruppen durchgeführt, siehe Kapitel 2.6.

Varianz der Paketlaufzeit

Da viele Anwendungen im Internet empfindlich auf eine große Varianz der Paketlaufzeit reagieren, möchte man auch diese quantitativ erfassen. Innerhalb der IPPM-Gruppe gibt es daher Bestrebungen, auch eine Größe für die Varianz der Paketlaufzeit zu definieren. Dieses Vorhaben ist noch nicht abgeschlossen und hat bislang noch nicht zu einem offiziellen RFC-Dokument geführt, es zeichnet sich aber ab, dass Definitionen im folgenden Sinne als sinnvoll erachtet werden.

Name: Varianz der Unidirektionalen Typ-P-Laufzeit („Type-P-One-way-ipdv“, IP delay variation).

Parameter:

A_1:	ein Host, dargestellt durch eine IP-Adresse
A_2:	ein Host, dargestellt durch eine IP-Adresse
T_1:	ein Zeitpunkt
T_2:	ein Zeitpunkt
P:	ein Pakettyp

Maßeinheit: Eine reelle Zahl von Sekunden (positiv, negativ oder Null), oder undefiniert.

Definition: Die reelle Zahl $\Delta\Delta T$ ist die *Varianz der unidirektionalen Typ-P-Laufzeit von A_1 nach A_2 zum Zeitpunkt T_1 und T_2* genau dann, wenn ΔT_1 die unidirektionale Typ-P-Laufzeit von A_1 nach A_2 zum Zeitpunkt T_1 und ΔT_2 die unidirektionale Typ-P-Laufzeit von A_1 nach A_2 zum Zeitpunkt T_2 und $\Delta\Delta T = \Delta T_2 - \Delta T_1$.

$\Delta\Delta T$ ist undefiniert, wenn mindestens eins von ΔT_1 oder ΔT_2 undefiniert ist.

Diskussion: Die unidirektionale Varianz der Paketlaufzeit hat gegenüber der Paketlaufzeit wegen der Differenzbildung den Vorteil, dass sie keine exakte Synchronisation der beteiligten Hosts benötigt. Die Ganggenauigkeit der Uhren beider Hosts ist jedoch auch hier essenziell.

Auch hier wird eine zusammengesetzte Größe definiert:

Name: Varianz einer Poisson-Sequenz unidirektionaler Typ-P-Laufzeit („Type-P-One-way-ipdv-Poisson-Stream“).

Parameter:
A_1: ein Host, dargestellt durch eine IP-Adresse
A_2: ein Host, dargestellt durch eine IP-Adresse
T_0: ein Zeitpunkt
T_f: ein Zeitpunkt
λ: eine Senderate in s^{-1}
P: ein Pakettyp

Maßeinheit: Eine Sequenz von Tripeln, wobei sich ein Tripel zusammensetzt aus T, einen Zeitpunkt, TI, einem Zeitintervall, und $\Delta\Delta T$, einer reellen Zahl von Sekunden oder undefiniert.

Die Sequenz von T ist monoton steigend. Bei jedem Paar sind T und $T + TI$ mögliche Parameter und $\Delta\Delta T$ ein mögliches Ergebnis der Varianz der unidirektionalen Typ-P-Laufzeit.

Definition: Es wird ein (Pseudo-)Zufalls-Prozess erzeugt, der im Intervall $[T_0, T_f]$ Poisson-verteilt Werte für T erzeugt. Sie werden als T_i mit $i = 1 \ldots n$ bezeichnet, wenn n die Anzahl der im Intervall liegenden Werte ist. Für jedes T_i mit $i = 1 \ldots n - 1$ wird die Varianz der unidirektionalen Typ-P-Laufzeit, $\Delta\Delta T_i$, bestimmt und bildet mit T_i und $TI = T_{i+1} - T_i$ ein Tripel der Ergebnis-Sequenz.

Diskussion: Auf der Basis diese Messreihe lassen sich wieder sinnvolle Statistische Größen aufbauen, wie zum Beispiel das inverse Perzentil („Type-P-One-way-ipdv-inverse-percentile“), das beschreibt, welcher Anteil aller Varianz-Werte einer Messreihe einen bestimmten Schwellwert nicht überschreitet (im Falle eines positiven Schwellwerts) beziehungsweise nicht unterschreitet (im Falle eines negativen Schwellwerts).

Der Absolutwert der Varianzwerte $|\Delta\Delta T|$ wird auch als **Jitter** („Type-P-One-way-ipdv-jitter“) bezeichnet.

3.3 Paketverlust

Paketverluste sind im Internet eine normale, alltägliche Erscheinung. Das Netzschicht-Protokoll IP übernimmt definitionsgemäß keine Garantie, dass ein Paket am Ziel-Host ankommt. Trotzdem oder gerade deswegen ist die Häufigkeit von Paketverlusten ein grundlegender Bestandteil der Qualität einer Internet-Verbindung:

- Viele Anwendungen arbeiten schlecht oder gar nicht, wenn die Paketverlustrate einen bestimmten Schwellwert überschreitet.
- Für viele Transportprotokolle ist die Paketverlustrate von unmittelbarer Bedeutung für ihre Performance: je kleiner die Paketverlustrate, desto besser der erzielbare Durchsatz.

- In der Regel sind Paketverluste ein Zeichen von Überlastungen irgendwo im betroffenen Pfad.

Die IPPM-Gruppe beschreibt in RFC 2680 [4] einige Definitionen für Maße des unidirektionalen Paketverlusts.

Unidirektionaler Paketverlust

Name: Unidirektionaler Typ-P-Paketverlust („Type-P-One-way-Packet-Loss“).

Parameter:

A_1:	ein Host, dargestellt durch eine IP-Adresse
A_2:	ein Host, dargestellt durch eine IP-Adresse
T:	ein Zeitpunkt
P:	ein Pakettyp

Maßeinheit: Boolean, dargestellt als 1 (Paketverlust) oder 0 (kein Paketverlust).

Definition: Der *unidirektionale Typ-P-Paketverlust von Host A_1 zu Host A_2 zur Zeit T* ist 0, wenn Host A_1 zum Zeitpunkt T ein Paket zu Host A_2 abschickt und dieses Paket irgendwann fehlerfrei dort ankommt. Falls das Paket nicht oder nicht fehlerfrei ankommt, ist der Paketverlust 1. Dabei zählt beim Absenden genau der Zeitpunkt der Übertragung des ersten Bit des Pakets.

Diskussion: Der Typ-P-Paketverlust ist immer dann 1, wenn die zugehörige Typ-P-Laufzeit undefiniert ist, und 0, wenn die zugehörige Typ-P-Laufzeit einen endlichen, definierten Wert hat.

Die Frage, ob die Ankunft von fehlerhaften Paketen bei Host A_2 als Paketverlust gewertet werden soll oder nicht, ist diskussionsfähig. Die Probleme der richtigen Zuordnung von fehlerbehafteten Paketen sind jedoch so schwerwiegend, dass die oben genannte Definition gewählt wurde. Außerdem sind wenige oder keine Internet-Anwendungen bekannt, die aus einem fehlerhaft empfangenen Paket Gewinn schöpfen können und es anders bewerten würden als einen Paketverlust.

Für eine Messung dieser Größe muss eine sinnvolle Obergrenze für die Laufzeit des Pakets gefunden werden. Wie bereits in Kapitel 3.1 beschrieben könnten dafür zwar theoretische Obergrenzen gefunden werden, in der Praxis wird man jedoch auf empirische Erfahrungen aufbauen müssen. Tatsächlich ist für die meisten Anwendungen und Transportprotokolle ein Paket mit einer Laufzeit über einem bestimmten Wert gleichwertig mit einem Paketverlust.

Name: Poisson-Sequenz unidirektionaler Typ-P-Paketverluste („Type-P-One-way-Packet-Loss-Poisson-Stream“).

Parameter:
- A_1: ein Host, dargestellt durch eine IP-Adresse
- A_2: ein Host, dargestellt durch eine IP-Adresse
- T_0: ein Zeitpunkt
- T_f: ein Zeitpunkt
- λ: eine Senderate in s^{-1}
- P: ein Pakettyp

Maßeinheit: Eine Sequenz von Paaren, wobei sich ein Paar zusammensetzt aus T, einen Zeitpunkt, und L, einem durch 0 oder 1 dargestellten Boolean-Wert.

Definition: Es wird ein (Pseudo-)Zufalls-Prozess erzeugt, der im Intervall $[T_0, T_f]$ mit der Rate λ Poisson-verteilt Werte für T erzeugt. Für jedes T wird der unidirektionale Typ-P-Paketverlust L bestimmt und bildet mit diesem T ein Paar aus der Ergebnis-Sequenz.

Diskussion: Es ist anzumerken, dass die so bestimmten Paketverluste durch die Poisson-Verteilung gut gegen Selbst-Beeinflussung geschützt ist im Gegensatz zu den von Transport-Protokollen beobachteten Paketverlusten. Beispielsweise treten TCP-Pakete meist gebündelt auf, was zu einer höheren Paketverlustrate während der Übertragung führen kann. Andererseits versucht TCP, Überlastsituationen zu erkennen und sich daran anzupassen, was wiederum zu niedrigeren Paketverlusten führt.

Name: Mittlerer unidirektionaler Typ-P-Paketverlust („Type-P-One-way-Packet-Loss-Average“).

Maßeinheit: Eine Reelle Zahl von 0 bis 1.

Definition: Basierend auf einer Poisson-Sequenz unidirektionaler Typ-P-Paketverluste wird der mittlere unidirektionale Typ-P-Paketverlust als arithmetisches Mittel der als Zahl aufgefassten L-Werte gebildet.

Diskussion: RFC 2680 weist darauf hin, dass in einer guten Internet-Verbindung die Paketverluste unter 1 % liegen. Wenn man Paketverluste von unter 1 % auflösen möchte bei gleichzeitig guter Auflösung von T, dann muss man einen Wert für λ wählen, der größer ist, als es aus anderen Gründen wünschenswert wäre (etwa wegen der Kosten für den Messverkehr oder die Gefahr, die Netzsituation durch den Messverkehr selber nennenswert zu beeinflussen).

Der mittlere unidirektionale Typ-P-Paketverlust dient im folgenden als Schätzwert für die *unidirektionale Paketverlustrate* p_u.

Bidirektionaler Paketverlust

Bislang hat die IPPM-Gruppe keine Definitionen für bidirektionale Paketverluste getroffen. Zu den unidirektionalen Größen analoge Definitionen für den *bidirektionalen Typ-P-Paketverlust von Host A_1 zu Host A_2 zur Zeit T* und die *Poisson-Sequenz unidirektionaler Typ-P-Paketverluste* liegen jedoch nahe. Wie schon bei der Paketlaufzeit werden diese Definitionen hier nicht mehr ausführlich dargestellt sondern als analog durchgeführt angenommen.

Diskussion: Auch hier gilt wieder, dass der Typ-P-Paketverlust immer dann 1 ist, wenn die zugehörige Typ-P-Laufzeit undefiniert ist, und 0, wenn die zugehörige Typ-P-Laufzeit einen endlichen, definierten Wert hat.

Die Durchführung von Messungen bidirektionaler Paketverluste ist wieder in dem Sinne einfacher als die unidirektionaler Paketverluste, dass keine spezielle Kooperation bei Host A_2 erforderlich ist, sondern dass unter Umständen das Standardverhalten beliebiger Hosts genutzt werden kann.

Der mittlere bidirektionale Typ-P-Paketverlust dient im folgenden als Schätzwert für die *bidirektionale Paketverlustrate* p_b.

Dabei ist die Formulierung „*bidirektionale Paketverlustrate zwischen Host A_1 und Host A_2*“ zulässig und gibt an, dass man die unidirektionalen Paketverlustraten für einen gewissen Zeitraum als konstant annimmt, so dass die genaue Reihenfolge der Untersuchung beider Richtungen vernachlässigt werden kann.

Relationen zwischen unidirektionaler und bidirektionaler Paketverlustrate

Die bidirektionale Paketverlustrate lässt sich aus den unidirektionalen Paketverlustraten herleiten:

$$p_b = 1 - (1 - p_{u1})(1 - p_{u2}) \tag{3.1}$$

$$= p_{u1} + p_{u2} - p_{u1}\,p_{u2} \tag{3.2}$$

mit p_b : bidirektionale Paketverlustrate zwischen A_1 und A_2
p_{u1} : unidirektionale Paketverlustrate zwischen von A_1 nach A_2
p_{u2} : unidirektionale Paketverlustrate zwischen von A_2 nach A_1

Umgekehrt lassen sich die unidirektionalen Paketverlustraten nicht ohne weiteres aus der bidirektionalen Paketverlustrate herleiten. In zwei Sonderfällen ist dies dennoch möglich:

Falls Paketverluste nur in einer Richtung auftreten und in der anderen Richtung gar nicht, also etwa $p_{u1} = 0$, dann gilt

$$p_{u2} = p_b \tag{3.3}$$

Falls die Paketverluste genau symmetrisch verteilt sind und in beiden Richtungen gleich häufig auftreten, also wenn $p_{u1} = p_{u2}$, dann lässt sich die einheitliche unidirektionale Paketverlustrate p_u wie folgt schreiben:

$$p_u = p_{u1} = p_{u2} = 1 - \sqrt{1 - p_b} \tag{3.4}$$

Die Annahme $p_{u1} = p_{u2}$ mag für den Normfall naheliegend sein, ist aber mit großer Vorsicht zu genießen. Wie viele der in der Praxis relevanten Internet-Pfade sich mit guter Näherung so beschrieben lassen und wie viele eindeutige Asymmetrien aufweisen, lässt sich nur schwer sagen und wird sich von Fall zu Fall deutlich unterscheiden.

3.4 Statistische Größen

Um kleinere mögliche Unstimmigkeiten zu vermeiden, definiert RFC 2330 ein paar grundlegende statistische Größen, die bei der Auswertung von Messergebnissen häufig verwendet werden.

Name: Empirische Verteilungsfunktion („empirical distribution function“).

Parameter: Eine Messreihe einer skalaren Größe x.

Definition: Die Funktion $F(x)$ ist die **empirische Verteilungsfunktion** einer gegebenen Messreihe von x, wenn $F(x)$ für jedes x angibt, welcher Anteil der gemessenen Werte kleiner oder gleich x ist. Ist x kleiner als jeder gemessene Wert, so ist $F(x) = 0$, ist x größer als jeder gemessenen Wert, dann ist $F(x) = 1$.

Diskussion: Die empirische Verteilungsfunktion wird im folgenden oft auch kurz **Verteilungsfunktion** genannt in dem Sinne, dass die empirische Verteilungsfunktion als Schätzwert für die tatsächliche Verteilungsfunktion verwendet wird.

Name: Perzentil („percentile“).

Parameter: Eine Messreihe einer skalaren Größe x.

Definition: Das i-te **Perzentil** einer gegebenen Messreihe ist der kleinste beobachtete Messwert x für den gilt $F(x) \geq i/100$.

Diskussion: Bei der Angabe eines Perzentil als Ergebnis einer Messung ist es stets wichtig, auch den Umfang N der Messreihe anzugeben, da sonst beispielsweise durch die Angabe unsinnig vieler Nachkommastellen ein falscher Eindruck der Genauigkeit erweckt werden kann.

Name: Median („median“).

Parameter: Eine Messreihe einer skalaren Größe x.

Definition: In der Stochastik bezeichnet der Median genau den Punkt x, für den die Wahrscheinlichkeiten, eine Beobachtung größer oder kleiner x zu machen, genau gleich groß sind. Der *Schätzwert für den Median*, der im folgenden auch kurz **Median** genannt wird, wird in Abhängigkeit des Umfangs N der Messreihe definiert:

Falls N gerade ist, ist der Median das 50-ste Perzentil gemäß obiger Definition.

Falls n ungerade ist, ist der Median das arithmetische Mittel der beiden zentralen Beobachtungen; mit $N = 2\,k$ also das arithmetische Mittel der k-t kleinsten und k-t größten Beobachtung.

Kapitel 4

Auswahl der zu untersuchenden Hosts

4.1 Motivation

Das heutige Internet hat eine Größe und Dynamik, bei der es unmöglich ist, eine Liste aller beteiligte Hosts zu erstellen oder sie auch nur abzuzählen. Jeder einzelne Benutzer mag sich in seinem Verhalten auf die Kommunikation mit einer überschaubaren Anzahl Hosts im Internet beschränken, aber sobald der Verkehr einer Gruppe von Benutzern gebündelt wird, steigt die Zahl der angesprochenen Hosts schnell an. Wenn nun die Verbindungsqualität zum „gesamten Internet“ untersucht werden soll, dann kann man dafür nur eine beschränkte Zahl Hosts nehmen, die repräsentativ für das gesamte Internet stehen. Diese Annäherung ist um so eher gerechtfertigt, je größer die Zahl dieser Hosts ist. Kapitel 4.2 untersucht am Beispiel der Gruppe der Internet-Benutzer der Universität zu Köln, auf wie viele verschiedene Internet-Hosts sich der Verkehr verteilt, und, in welcher Größenordnung eine repräsentative Auswahl von Internet-Hosts demnach liegen sollte.

Des weiteren spielt die Art der Auswahl der Hosts eine Rolle: wenn die Auswahl *objektiv* erfolgen soll, muss sie nach klar definierten Regeln erfolgen, mithin also *automatisierbar* sein. Nur eine automatisierte Erstellung dieser Hostliste bietet Schutz vor absichtlicher oder unabsichtlicher Beeinflussung der Ergebnisse. Als einzig sinnvoller Anhaltspunkt für die Auswahl von Hosts bietet sich eine *Analyse des tatsächlichen Internet-Verkehrs* einer gegebenen Benutzergruppe an: die Hosts, mit denen die Benutzer am meisten Daten austauschen, spielen die größte Rolle und sollten somit als erste in diese Liste aufgenommen werden. Neben dem Zugangspunkt, von dem aus die Internet-Konnektivität gemessen werden soll, ist also die Benutzergruppe, nach deren Vorlieben und Maßstäben beurteilt werden soll, ein weiterer Parameter der Messungen. Kapitel 4.3 beschreibt mehrere Möglichkeiten, das Benutzerverhalten in geeigneter Weise zu untersuchen, sowie

die durchgeführte Implementierung mit Hilfe von NetFlow-Accounting. In Kapitel 4.4 wird schließlich auf Probleme eingegangen, die sich bei groß angelegten Messungen in der Praxis ergeben haben, und es wird dargestellt, wie sie mit geeigneten Maßnahmen gelöst oder zumindest gemildert werden können.

4.2 Verteilung des Verkehrsaufkommens auf Hosts

Um zu wissen, wie viele Hosts man überhaupt untersuchen muss um eine Datenbasis zu erhalten, die für das ganze Internet repräsentative Aussagen erlaubt, wird im folgenden die Verteilung des Internet-Verkehrsaufkommens auf die wichtigsten Hosts untersucht.

Die folgenden Daten beziehen sich auf den im gesamten Monat Mai 1997 an der Universität zu Köln gemessenen Internet-Verkehr. Dabei wurde gezählt, wie viele Bytes mit Server-Rechnern außerhalb der Universität ausgetauscht wurden.

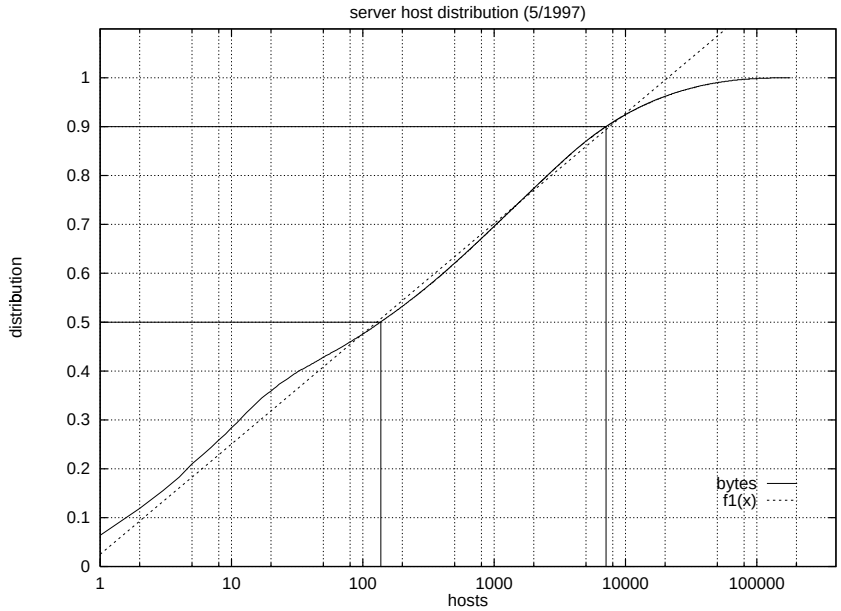

Abbildung 4.1: Verkehrsverteilung.

Aus Abbildung 4.1 lässt sich ablesen, welchen Anteil y die x wichtigsten Hosts am Gesamtverkehr haben. So ergibt sich beispielsweise, dass die ersten 132 Hosts 50 % des Gesamtverkehrs ausmachen. Um 90 % des Gesamtverkehrs zu erschließen, muss man allerdings schon 7113 Hosts untersuchen.

Wie man sieht, gibt es einen weiten Bereich, indem die Kurve gut durch eine Gerade angenähert werden kann. Wegen des logarithmischen Maßstabs

bedeutet das, dass der erfasste Anteil der Verkehrsaufkommens logarithmisch mit der Zahl der erfassten Hosts steigt. Eine Regressionsanalyse der Verteilung für die ersten 10000 Hosts ergibt folgende Näherung:

$$F(x) \approx 0.024997 + 0.09799 \ln(x) \tag{4.1}$$

Oder in anderen Worten: eine Verzehnfachung der Zahl der untersuchten Hosts steigert den erfassten Verkehrsanteil um etwa 22.5 %. Mit ungefähr 20000 Hosts wäre demnach der gesamte Verkehr erfasst. In der Praxis ist es aber so, dass sich die letzten 10 % des Verkehrsaufkommens auf sehr viele Hosts verteilen, die jeweils nur einen sehr kleinen Anteil haben.

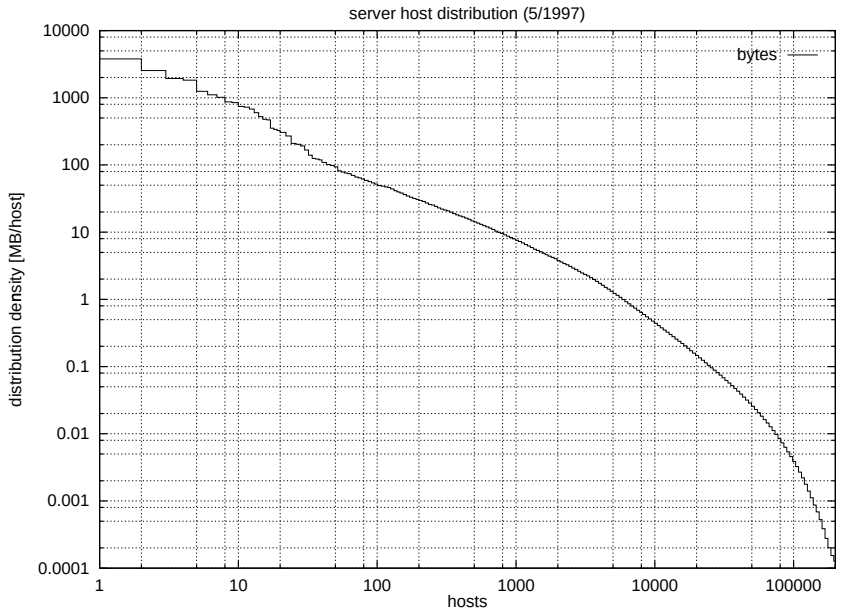

Abbildung 4.2: Verteilungsdichte des Verkehrsaufkommens.

Dazu zeigt Abbildung 4.2 noch einmal den Beitrag (in Megabytes pro Host und Monat), den die verschiedenen Hosts zum Gesamtverkehr liefern.

4.3 Implementierung

4.3.1 Werkzeuge zur Verkehrsanalyse

Im einfachsten Fall verwenden alle Teilnehmer der zu untersuchenden Gruppe ein geteiltes Netzmedium wie zum Beispiel Ethernet, FastEthernet oder FDDI. Dann ist es möglich, auf einem Host mit Programmen wie `tcpdump`

[51] oder `snoop`[1] alle Datenpakete aller Teilnehmer zu untersuchen und Statistiken darüber zu erstellen, welcher Internet-Host wie oft angesprochen wird. Falls beispielsweise durch die Verwendung von Ethernet-Switching ein exklusives Netzmedium zum Einsatz kommt, ist dieser Ansatz immer noch möglich, falls der Ethernet-Switch einen sogenannten **Span-Port** zu Verfügung stellt, der Kopien aller Pakete zu einem bestimmten Ausgang schickt.

Schwieriger wird es, wenn die Benutzergruppe über mehrere auch logisch getrennte Netze verteilt ist. Beispielsweise hat das Datennetz der Universität zu Köln (ohne Verwaltung und medizinische Einrichtungen) gegen Ende 1999 über 170 sogenannte Subnetze. Dann muss in jedem Netz ein solcher Host installiert werden, der dort den Verkehr beobachtet. Die Daten müssen zusammengeführt und addiert werden, ohne allerdings solche Pakete, die durch mehrere Subnetze laufen, mehrfach zu zählen. In einem solchen Fall hilft die inzwischen abgeschlossene Arbeit der IETF-Arbeitsgruppe Realtime-Traffic-Flow-Measurement (RTFM) [15], die eine Architektur von verteilten „Meter“, „Meter Reader“ und eines zentralen „Manager“ definiert. Dazu gibt es mit `NeTraMet` [16] bereits eine frei verfügbare Implementierung.

Wenn die Benutzergruppe über Subnetze verteilt ist, aber eine gemeinsame, breitbandige Verbindung zum Internet verwendet, dann lassen auch dort die benötigten Daten gewinnen. So verwendet das Projekt OCxmon [7] halbdurchlässige Spiegel, mit denen aus einer Glasfaserleitung ein kleiner Teil des Lichts ausgekoppelt wird, um mit umgebauter, handelsüblicher Hardware den Datenverkehr zu untersuchen.

Eine andere Möglichkeit, die Messungen zu vereinfachen, ergibt sich, wenn die Router selber in der Lage sind, die relevanten Daten zu erfassen. Dazu kann man in der Regel in einem Router das sogenannte **IP-Accounting** aktivieren. Der Router pflegt dann in einer Matrix-artigen Datenstruktur für jede beobachtete Kombination aus Absende- und Ziel-IP-Adresse einen Zähler für die Zahl der Pakete und Byte. Diese Zähler können dann von Zeit zu Zeit ausgelesen und zurückgesetzt werden. Aus dem IP-Accounting wird zwar deutlich, mit welchen Internet-Hosts der meisten Verkehr abgewickelt wird, es hat aber zwei wesentliche Nachteile:

- Die gesammelten Daten beinhalten keinerlei Informationen über die in den gezählten Paketen verwendeten Protokolle (TCP, UDP, ...) und Dienste (gegeben durch UDP- oder TCP-Portnummer). Diese Informationen sind aber beispielsweise für das in Kapitel 6 vorgestellte Messverfahren wichtig.

- Die Pflege von Zählern von jeder Kombination aus Absende- und Ziel-IP-Adresse verbraucht viel Ressourcen im Router, vor allem Hauptspeicher. Zudem ist sie abhängig von der Art der Benutzung, so dass ei-

[1] `snoop` ist ein Bestandteil des Unix-Betriebssystems Solaris der Firma SUN Microsystems.

ne ungewöhnliche Benutzung diesen Ressourcenverbrauch ungewöhnlich stark anschwellen lassen kann, so dass sogar Router, die normalerweise gut für IP-Accouting gerüstet sind, überfordert werden. So kam es beispielsweise schon vor, dass große Teile des vom DFN e.V. betriebenen deutschen Wissenschaftsnetzes ausfielen, weil einzelne Benutzer sogenannte **Hostscans** und **Portscans** durchgeführt hatten — Verfahren, bei denen ungewöhnlich viele verschiedene IP-Adressen beziehungsweise Portnummern angesprochen werden und die typischerweise bei den Vorbereitungen elektronischer Angriffe zum Identifizieren möglicher Opfer und vorhandener Schwachstellen dienen.

Vor allem aus dem letztgenannten Grund aktiviert man IP-Accounting nicht gerne in Routern, die stark aggregierten Verkehr führen.

Eine interessante Alternative zum IP-Accounting ist das **NetFlow-Accouting** [82], bei dem nicht nur Absende- und Ziel-IP-Adressen, sondern auch Protokoll und gegebenenfalls TCP/UDP-Portnummer unterschieden werden. Im Rahmen von sogenannten **Flows** werden dabei die Zahl der Pakete und Byte aufsummiert und zu einem Rechner übertragen, der daraus beliebige Statistiken erzeugen kann.

Mittlerweile lässt sich NetFlow-Accouting mit `NeTraMet` kombinieren, so dass sich auch bei komplexeren Strukturen mit mehreren verteilten Routern effizient die benötigten Daten gewinnen lassen.

4.3.2 Implementierung der Verkehrsanalyse mittels NetFlow-Accounting

An der Universität zu Köln gibt es einen zentralen Router, an dem praktisch alle Verbindungen der Universität mit dem Internet konzentriert sind. Auf diesem Router ist NetFlow-Accouting aktiviert, so dass sich daraus bereits alle gewünschten Statistiken gewinnen lassen.

Auswahlregel

Zunächst muss eine klare Regel festgelegt werden, aus welchen Hosts in die Liste zusammengesetzt werden soll. Sie wurde wie folgt gewählt:

> Es werden die 1000 Server-Hosts außerhalb der Universität bestimmt, mit denen die Benutzer der Universität die meisten Daten austauschen.

Die Zahl von 1000 Hosts wurde gewählt, weil damit gemäß Abbildung 4.1 immerhin zwei Drittel des gesamten Verkehrsaufkommens repräsentiert werden können und weil sich damit die Messungen noch im machbaren Rahmen halten.

Des weiteren wird hier auf das Client/Server-Prinzip (siehe Kapitel 6.2) bezug genommen. Indem hier nach *Server*-Hosts gesucht wird, wird vor allem die aus Client-Sicht interessante Konnektivität zum Internet betrachtet. Das ist deshalb sinnvoll, weil die Internet-Benutzer der Universität überwiegend Client-seitig auf Angebote im Internet zugreifen. Aus Sicht eines reinen Dienst-Anbieters würde man diese Regel eventuell anders aufstellen.

Allerdings hätte die Untersuchung reiner Clients im Internet zusätzliche Probleme: anders als Server haben Clients nicht immer statische IP-Adressen und selbst dann werden sie oft nach Benutzung ausgeschaltet, so dass sie nur sehr eingeschränkt zu Verfügung stünden. Darüber hinaus werden im Internet zunehmend Schutzmechanismen („Firewall“, „Screening Router“) eingesetzt, die Client-Hosts gänzlich vor aktiven Zugriffen von außen schützen[2]. In anderen Worten: während Server generell für Messungen von außen zu Verfügung stehen, gilt dies in zunehmendem Maße für Clients nicht mehr. Für eine Server-Sicht auf das Internet wären Verfahren zu entwickeln, bei denen Clients im Internet gezählt werden, für die Messungen aber beispielsweise ersatzweise jeweils solche Server angesprochen werden, die den entsprechenden Clients topologisch möglichst nahe liegen. Oder man begnügt sich mit einem Ansatz wie dem in Kapitel 2.7 dargestellten.

Bestimmung von Server-Ports

Typischerweise unterscheidet sich bei UDP und TCP die Server- von der Client-Seite dadurch, dass die Server-Seite eine feste Portnummer verwendet (zum Beispiel TCP-Port 80 für HTTP, TCP-Port 23 für telnet und UDP-Port 53 für DNS; genaueres in Kapitel 6.2). Die Client-Seite hingegen verwendet für jede neue Verbindung eine andere Portnummer, die vom jeweiligen Betriebssystem zufällig oder nach einem bestimmten Muster vergeben wird.

Im ersten Ansatz wurden alle bekannten Server-Portnummer als Server-Ports und alle anderen als Client-Ports gewertet. Dabei zeigte sich allerdings, dass neben den offiziell dokumentierten Server-Ports [26] auch viele weitere Portnummer für Server verwendet werden. Der Versuch, auch diese zu erkennen, scheiterte daran, dass viele Dienste unter verschiedenen Portnummern angeboten werden (beispielsweise HTTP und IRC) und dass immer wieder neue Dienste unter neuen Portnummern entstehen.

Außerdem kommt es vor, dass bestimmte Portnummern sowohl auf Server-Seite, als auch recht häufig auf Client-Seite eingesetzt werden. Dadurch kann es zu Uneindeutigkeiten bei der Bestimmung der Server-Seite einer Verbindung kommen.

Als Ausweg wurde ein automatisiertes Verfahren enwickelt, das UDP- und TCP-Portnummern in ihrer *relativen* Bedeutung als Server-Port bewertet. Diese Bewertung wird täglich neu durchgeführt, so dass das Aufkommen

[2]Die Universität zu Köln betreibt einen solchen Mechanismus seit 1999 [41].

neuer Dienste genauso berücksichtigt wird wie das Zurückgehen früherer Dienste. Anhand dieser relativen Bedeutung als Server-Port können dann in TCP- und UDP-Verbindungen Client- und Server-Seite zugeordnet werden.

Die relative Bedeutung als Server-Port ergibt sich aus Menge Flows, die eine Portnummern auf sich zieht. Ein Flow wird dabei definiert als ein Tupel $(A_{src}, A_{dst}, prot, port_{src}, port_{dst}, N_{pkt}, N_{oct})$ mit

A_{src} : IP-Adresse des absendenden Hosts,
A_{dst} : IP-Adresse des Ziel-Hosts,
$prot$: Nummer des Transportprotokolls (TCP, UDP, ICMP, ...),
$port_{src}$: Portnummer des Transportprotokolls auf Sender-Seite,
$port_{dst}$: Portnummer des Transportprotokolls auf Empfänger-Seite,
N_{pkt} : Zahl der Pakete in diesem Flow und
N_{oct} : Zahl der Byte (Oktette) in diesem Flow.

Dabei sind $port_{src}$ und $port_{dst}$ nur definiert, wenn das Transportprotokoll TCP oder UDP ist. Nur für diese Protokolle werden Server-Ports untersucht.

Zur Bestimmung der relativen Bedeutung als Server-Port werden symmetrische Matrizen B aufgebaut, deren Elemente $B_{port_{src},port_{dst}} = B_{port_{dst},port_{src}}$ die Summe aller beobachteter Flows zwischen zwei Ports enthalten. Die Übertragungsrichtung wird dabei nicht unterschieden. Es wird eine Matrix B_{TCP} für den TCP- und eine Matrix B_{UDP} für den UDP-Verkehr aufgebaut. Ihre Elemente werden auf 0 initialisiert und anschließend kommt der in Abbildung 4.3 dargestellte Algorithmus zum Einsatz. S_{port} ist dabei die relative Bedeutung als Server-Port für den Port $port$.

- Für jeden Flow $(A_{src}, A_{dst}, prot, port_{src}, port_{dst}, N_{pkt}, N_{oct})$:
  - Falls $prot$ = TCP:
    - $B_{TCP\,port_{src},port_{dst}} \leftarrow B_{TCP\,port_{src},port_{dst}} + 1$
    - $B_{TCP\,port_{dst},port_{src}} \leftarrow B_{TCP\,port_{src},port_{dst}}$
  - Falls $prot$ = UDP:
    - $B_{UDP\,port_{src},port_{dst}} \leftarrow B_{UDP\,port_{src},port_{dst}} + 1$
    - $B_{UDP\,port_{dst},port_{src}} \leftarrow B_{UDP\,port_{src},port_{dst}}$
- Für $B = B_{TCP}, B_{UDP}$:
  - $z \leftarrow Z_{max}$
  - Solange B noch Spalten und Zeilen hat und $z > 0$:
    - Bestimme m so, dass $S_m = \sum_i B_{m,i}$ maximal wird.
    - Ausgabe von (m, S_m).
    - Entferne die m-te Spalte und Zeile aus B.
    - $z \leftarrow z - 1$

Abbildung 4.3: Algorithmus zur Bestimmung der relativen Bedeutung als Server-Port.

Die Vorgabe eines Maximalwerts Z_{max} für die Zahl der ermittelten Server-Ports ist notwendig, weil das Verfahren sonst eine starke Abhängigkeit vom Benutzerverhalten hätte: die gleichen Portscans, die dem IP-Accounting in Routern zu schaffen machen, würden auch dieses Verfahren überfordern. Z_{max} wurde für die laufenden Messungen zu 2000 gewählt. Dies ist ausreichend, weil ohnehin nicht mehr als 2000 verschiedene Server untersucht werden sollen.

Das Verfahren wurde in der Programmiersprache Perl [87] implementiert. Die Matrizen B können im Prinzip sehr groß werden, weil Portnummern beliebig im Bereich von $1 \ldots (2^{16} - 1)$ liegen können. Durch eine Implementierung mit Hilfe von Hash-Tabellen ließ sich dieses Problem lösen. Die wiederholte Neuberechnung aller möglicher Summen $S_m = \sum_i B_{m,i}$ wäre sehr aufwändig; um das Verfahren effizient zu halten, wurden Hilfszähler implementiert, die am Anfang aufgebaut und mit jeder Entfernung von Spalten und Reihen dekrementiert werden.

Die Ausgabe des Programms sieht dann so aus (stark gekürztes Beispiel, hier die ersten 20 Ports für das Protokoll TCP):

```
80      8845459
25      322549
21      250593
113     143071
9004    109771
20      88380
443     81877
110     60266
8080    41668
23      32044
53      29805
119     26936
8084    13221
6667    7944
9065    7741
3333    6709
22      6706
81      6459
6207    6278
8000    6068
```

In der erste Spalte findet die TCP-Portnummer und in der zweiten die zugehörige relative Bedeutung, die Anzahl der beobachteten Flows.

Bestimmung der Server-Hosts

Die Bestimmung der Server-Hosts erfolgt nach dem Algorithmus in Abbildung 4.4. U_K sei dabei die Menge der IP-Adressen der Universität zu Köln und $C_{TCP,A,port}$ beziehungsweise $C_{UDP,A,port}$ ein Zähler für die Datenmenge, die der Server mit der IP-Adresse A auf seinem Port *port* für die Universität zu Köln verarbeitet hat.

- Für jeden Flow $(A_{src}, A_{dst}, prot, port_{src}, port_{dst}, N_{pkt}, N_{oct})$:
  - Falls $A_{src} \in U_K \wedge A_{dst} \notin U_K \wedge S_{port_{src}} \leq S_{port_{dst}}$:
    - $C_{prot,A_{dst},port_{dst}} \leftarrow C_{prot,A_{dst},port_{dst}} + N_{oct}$
  - Falls $A_{src} \notin U_K \wedge A_{dst} \in U_K \wedge S_{port_{src}} \geq S_{port_{dst}}$:
    - $C_{prot,A_{src},port_{src}} \leftarrow C_{prot,A_{src},port_{src}} + N_{oct}$
- Für $prot = TCP, UDP$:
  - Für alle A mit $\sum_{port} C_{prot,A,port} > 0$, sortiert nach ebendieser Summe:
    - Bestimme m so, dass $C_{A,m}$ maximal wird.
    - Ausgabe von $\left(A, m, \sum_{port} C_{prot,A,port}\right)$.

Abbildung 4.4: Algorithmus zur Bestimmung der Server-Hosts.

Auch dieses Programm wurde in Perl implementiert. Seine Ausgabe sieht wie folgt aus (stark gekürztes Beispiel):

```
207.87.19.48     80      6068254 6904550891      38761
128.176.107.90   20      1955724 1790272535      849
207.204.214.71   80      1317292 1055572985      1356
193.174.144.182  20      520322  778733872       661
209.1.224.15     80      749520  636346679       12764
209.1.224.12     80      745316  571050751       20673
134.93.196.106   6000    1907796 565154372       552
137.226.144.2    8080    459636  462509276       11087
209.1.224.14     80      595543  436404349       10893
130.75.178.41    20      425332  392580172       521
```

Zur Information wird hier in jeder Zeile neben der IP-Adresse des Hosts und der Portnummer auch die Anzahl der gezählten Pakete, Byte und Flows ausgegeben.

Um regelmäßige Schwankungen in der Nachfrage der Benutzer auszugleichen und um beispielsweise nicht immer nur montags diejenigen Server

zu untersuchen, die speziell an Wochenenden beliebt sind, wird diese Liste zwar täglich erzeugt, es werden aber immer die Listen der vergangenen sieben Tage zusammengefasst.

4.4 Ausnahmeregelungen

4.4.1 Motivation

Bei der Auswahl der zu untersuchenden Hosts wird neben den oben genanntem Verfahren noch eine Ausnahmeliste berücksichtigt: eine Liste von Hosts, zu denen auf keinen Fall Messungen durchgeführt werden sollen.

In diese Ausnahmeliste werden Hosts aus einem der beiden folgenden Gründe eingetragen:

- Der Host ist durch Schutzmechanismen wie einer sogenannten Firewall oder einem Screening Router gegen unerwünschte Zugriffe geschützt und dabei werden auch die verwendeten Messpakete geblockt. Der Host ist damit für die Messungen unbrauchbar.

  In vielen Fällen gibt sich ein solcher Mechanismus zu erkennen, indem er an den Absender der Messpakete einzelne ICMP-Pakete vom Typ „communication administratively prohibited“ schickt. Diese ICMP-Pakete werden in den laufenden Messungen mittlerweile erkannt und setzen den betroffenen Ziel-Host automatisch auf die Ausnahmeliste.

- Eine solche Firewall erkennt die Messpakete als unerwünschten Verkehr, verschickt aber keine ICMP-Pakete an deren Absender, sondern protokolliert die Messpakete stillschweigend – oftmals mit dem Kommentar „potenzieller Angriff“. Ein so alarmierter Administrator wendet sich nun persönlich an den Absender der Messpakete und verlangt das Abstellen der Messungen.

Die in den Kapitel 5 und 6 vorgestellten Messverfahren sind durchweg so konzipiert, dass kein tatsächlicher Angriff stattfindet und dass Sicherheit und Leistungsfähigkeit der betroffenen Ziel-Hosts nicht eingeschränkt werden; aber die recht einfachen Regeln von Firewall-Systemen können die Messpakete oft nicht von möglichen echten Angriffen unterscheiden.

4.4.2 Beschwerdemöglichkeiten

Im Zeitraum von 1998 bis Januar 2000 wurden 25 Beschwerden über die laufenden Messungen empfangen. Dabei haben die Adminstratoren sehr unterschiedliche Möglichkeiten verwendet, ihre Beschwerde an den Absender der Pakete zu adressieren:

- Das Domain-Name-System (DNS) ordnet IP-Adressen verschiedene Attribute zu, im wesentlichen Namen für Hosts und Netze. Unter anderen wird dabei auch jedem Bereich von IP-Adressen ein sogenannter „Start-of-Authority (SOA) Record“ zugewiesen, der verbindlich eine E-Mail-Adresse für die Kontaktperson des Adressbereichs enthält.

  Im Fall der Universität zu Köln führt diese E-Mail-Adresse über eine Mailingliste im Regelfall direkt zum Verursacher der Messungen und hat im Falle seiner Abwesenheit eine geordnete Vertreterregelung.

- Oft wird der Unix-Standard-Account `root` auf dem die Messpakete versendendem Host adressiert. Dies gelangt im Fall der Universität zu Köln auf dem Umweg über informierte Kollegen zum Ziel.

- RIPE [71] betreibt eine öffentlich zugängliche Datenbank, in der unter anderem technische und administrative Kontaktpersonen für jeden IP-Bereich angegeben werden.

  Im Fall der Universität zu Köln sind alle technischen Kontaktpersonen über die Natur der Messungen informiert und konnten schnell weiterhelfen. Die administrative Kontaktperson ist der geschäftsführende Direktor des Rechenzentrums; an diese Kontaktperson adressierte Beschwerden belasteten für ihre Bearbeitung also zusätzlich das Direktorium des Rechenzentrums.

- In einem bislang einzigen Fall wurde auf die Konsultierung jeglicher technischer Netz-Datenbanken verzichtet und eine Beschwerde direkt an den Rektor der Universität adressiert. Diese Beschwerde belastete das Rektorat der Universität, das Direktorium des Rechenzentrums und musste dementsprechend mit noch größerem Aufwand bearbeitet werden.

- In einem anderen Einzelfall wurde die erklärende WWW-Seite (siehe Abbildung 4.5) zwar gefunden, aber so gründlich missverstanden, dass die IETF-Arbeitsgruppe IPPM fälschlicherweise für den Verursacher der Messungen gehalten wurde und eine Beschwerde zusätzlich an die Leiter der Arbeitsgruppe sowie an den Direktor der „Transport Area“ gerichtet wurde.

In allen Fällen wurden die erhobenen Vorwürfe nach einer kurzen Erklärung der Messungen zurückgenommen. In einigen Fällen wurden die Messungen dann explizit erlaubt, in anderen Fällen wurde wegen Schwierigkeiten mit der Protokollierung eine Einstellung der Messungen erbeten.

4.4.3 Maßnahmen

Um bei Beschwerden den Aufwand für beide Seiten möglichst gering zu halten, wurden im Laufe der Zeit mehrere Vorkehrungen getroffen:

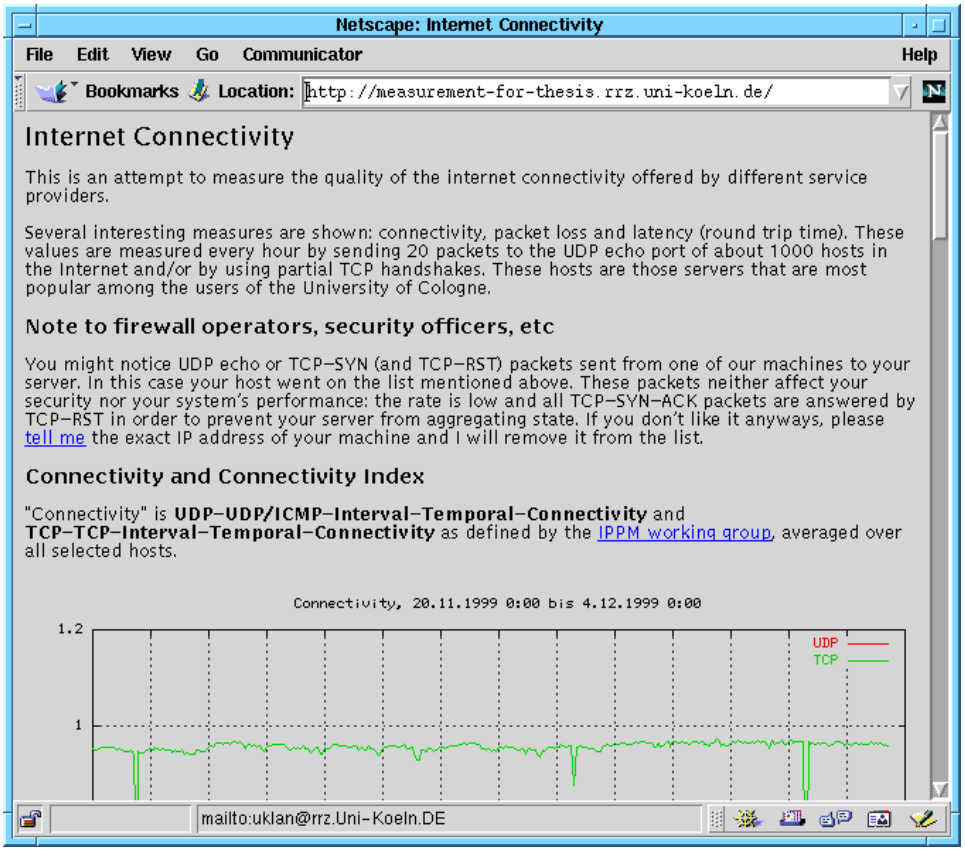

Abbildung 4.5: WWW-Seite zur Erläuterung der Messungen.

- Es wurde eine WWW-Seite eingerichtet, die die Messungen kurz erläutert, Administratoren auf die günstigste Kontaktadresse hinweist und nebenbei aktuelle Statistiken anzeigt, siehe Abbildung 4.5. Die reine Existenz dieser WWW-Seite nützt natürlich noch nichts, wenn ein Betroffener nicht auf sie hingewiesen wird.

- Für die UDP-basierten Messungen wurde daher der Datenteil der erzeugten Pakete für einen solchen Hinweis genutzt. Allerdings scheinen die wenigsten Administratoren verdächtige Pakete so genau zu untersuchen: nach mehreren Jahren fortwährender Messungen gab es genau eine E-Mail, aus der zu schließen war, dass ein Administrator diesen Hinweis überhaupt entdeckt hatte.

- Im DNS wurde für den Mess-Host ein Record vom Typ TXT angelegt, der explizit auf diese WWW-Seite hinweist. Dies hatte praktisch gar keinen Effekt. Offenbar sucht bei der Analyse eines verdächtigen Hosts niemand nach DNS-Einträgen diesen Typs.

- Für die Messungen wurde exklusiv eine IP-Adresse festgelegt, der ein Name zugeordnet wurde, der eindeutig auf den Charakter der Messungen hinweist: `measurement-for-thesis.rrz.uni-koeln.de`.
- Unter dieser Adresse wurde ein HTTP-Server eingerichtet, der direkt auf die oben genannte WWW-Seite verweist.

Vor allem die beiden letztgenannten Maßnahmen haben schließlich dazu geführt, dass die Beschwerden in mehr als der Hälfte aller Fälle an die gewünschte Kontaktadresse gerichtet wurden und an Schärfe verloren.

In diesem Zusammenhang wird gelegentlich gefordert, *vor* der Durchführung der Messungen das Einverständnis der Gegenseite einzuholen. Auf der einen Seite wäre dies wünschenswert, auf der anderen Seite läuft aber das individuelle Einholen solcher Einverständnisse der beabsichtigen *Objektivität* bei der Auswahl der Hosts zuwider. Außerdem ist die Notwendigkeit eines solchen Einverständnisses nicht wirklich gegeben, da nur öffentlich dokumentierte Protokolle verwendet werden und keineswegs tatsächlich Angriffe auf die Sicherheit oder Leistungsfähigkeit anderer Systeme durchgeführt werden. Darüber hinaus würde das bloße Versenden einer E-Mail an jeden betroffenen Ziel-Host wegen der großen Zahl der zu untersuchenden Hosts auf beiden Seiten mehr Arbeit und damit Schaden verursachen, als die Beantwortung vergleichsweise seltener Beschwerden oder Rückfragen.

4.4.4 Analyse – Anteil erreichbarer Hosts

In der Zeit von Juni 1998 bis Januar 2000 wurden kontinuierlich stündliche Messungen zu 1000 bis 2000 verschiedenen Hosts durchgeführt. Während des gesamten Zeitraums wurden UDP-basierte Messungen durchgeführt (siehe Kapitel 5); im Sommer 1998 sowie seit Juli 1999 zusätzlich TCP-Syn-basierte Messungen (siehe Kapitel 6). Während dieser Zeit gingen auf den verschiedenen Wegen insgesamt 25 Beschwerden ein. Von diesen Beschwerden bezogen sich 21 auf die UDP-basierten Messungen und 4 auf die TCP-Syn-basierten.

	Anzahl Hosts	Verkehrsanteil [%]
Verboten	431	0.284
Gefiltert	70	0.839
Erreichbar	3988	67.824

Tabelle 4.1: Anteil verfügbarer/nicht-verfügbarer Hosts

Auf Grundlage des Internetverkehrs der Universität zu Köln in der Woche vom 4. bis 10.12.1999 zeigt Tabelle 4.1 die Anzahl der Hosts, die für die laufenden Messungen zu Verfügung standen beziehungsweise nach den langfristig aufgebauten Ausnahmelisten nicht zu Verfügung standen. Außerdem

wird aufgeführt, welchen Anteil diese Hosts am gesamten Internetverkehrs der Universität haben. Als „verboten“ werden die Hosts bezeichnet, die aufgrund einer Beschwerde in die Ausnahmeliste aufgenommen wurden. Da sich einige Beschwerden auf ganze Netze und nicht nur auf einzelne Hosts bezogen, ist die Zahl der verbotenen Hosts größer als die der Beschwerden. Als „gefiltert“ werden solche Hosts bezeichnet, die wegen eines Paketfilters nicht erreichbar waren und bei deren Untersuchung entsprechende ICMP-Unreachable-Pakete empfangen wurden. Die Zahl der insgesamt in der Woche erreichten Hosts ist größer als die der jede Stunde untersuchten Hosts, da die Hostliste entsprechend der genannten Verfahren dynamisch erzeugt wird und gewisse Schwankungen aufweist.

Mit 1.123 % Verkehrsaufkommen ist der Anteil der aus administrativen Gründen nicht verwendbaren Hosts recht gering. Und mehr als 67 % des Verkehrsaufkommens werden von den in der Woche erreichten Hosts abgedeckt.

Kapitel 5

Messungen mit UDP-Echo-Paketen

5.1 Motivation

Der erste, naheliegendste Ansatz, Konnektivität, Paketlaufzeit und -verluste zu messen, ist in der Regel die Verwendung von `ping`, einem Standardprogramm jedes IP-fähigen Betriebssystems. Die in den Kapiteln 2.2, 2.3 und 2.4 vorgestellten Projekte gehen so vor.

Im Prinzip ist `ping` eine Implementierung zur Messung der von der IPPM-Gruppe definierten Maße in den bidirektionalen Varianten. Als „Typ P“ kommt dabei ICMP-Echo-Request/ICMP-Echo-Reply zum Einsatz. Allerdings verwendet das Programm konstante Zeitabstände zwischen dem Durchführen mehrere Einzelmessungen und nicht die von der IPPM-Gruppe geforderte Poisson-Verteilung. Des weiteren sind in diesem Programm keine Vorkehrungen getroffen, zu mehr als einem Host Messungen durchzuführen. Da im Rahmen der vorliegenden Arbeit sowohl Messungen zu Tausenden von Hosts effizient ermöglich werden sollen als auch eine korrekte Durchführung von Messungen für die IPPM-Poisson-Sequenzen, musste auf jeden Fall eine Neuimplementierung durchgeführt werden.

Eine weitere Frage ist, ob die Pakettypen ICMP-Echo-Request/ICMP-Echo-Reply gut für die Bestimmung von Performance-Maßen für normalen Verkehr geeignet ist. Wie aus der Literatur [73] und Aussagen von Netzbetreibern [21] hervorgeht, wird ICMP heutzutage im Internet häufig anders behandelt als andere Pakettypen. Daher sollten Messungen möglichst mit dem Pakettypen UDP oder TCP arbeiten.

5.2 Erster Ansatz: Implementierung im User-Level

Im folgenden werden einige Untersuchungen vorgestellt, die die allerersten Erfahrungen mit groß angelegten Messungen darstellen. Die in Kapitel 4

beschriebenen Erkenntnisse und Methoden existierten zu dieser Zeit noch nicht.

Im ersten Ansatz wurden die Funktionen gängiger Programmierschnittstellen verwendet, um mit Hilfe des UDP-Echo-Dienstes schmalbandige Messungen von Paketlaufzeiten und Paketverlusten in der Netzschicht durchzuführen. Die Verwendung des UDP-Echo-Ports bietet sich zum einen wegen der einfachen Implementierung an. Zum anderen werden so unauffällige Internet-Pakete generiert, die von Netzbetreibern mit üblicher Priorität weitergereicht werden.

5.2.1 Implementierung

Zur Durchführung der Messungen und Untersuchung der Meßmethode wurde das Tool `jping` entwickelt. Wegen der guten Unterstützung von Multithreading wurde die Implementierung in der Programmiersprache Java vorgenommen.

Die Paketlaufzeit und Paketverlustrate werden ermittelt, indem UDP-Pakete an andere Rechner im Internet verschickt werden. Dabei wird der Echo-Port (Nr. 7, siehe [70]) angesprochen, was die Gegenstelle dazu veranlasst, den Datenteil des Pakets unmittelbar an den Absender zurückzuschicken [64]. Im Vergleich zu Benchmarks auf Anwendungsebene geschieht das mit minimaler Verzögerung, da der UDP-Echo-Dienst in der Regel direkt vom Prozess `inetd` erbracht wird und keine Rechner-interne Kommunikation mit anderen User-Level-Prozessen benötigt. Die Bestimmung der Paketlaufzeit erfolgt mit dem in Abbildung 5.1 gezeigten Java-Codezeilen.

```
DatagramPacket p = new DatagramPacket(buf,len,dest,port);
socket.send(p);
long send_time = System.currentTimeMillis();
...
socket.receive(p);
long recv_time = System.currentTimeMillis();
long rtt = recv_time - send_time;
```

Abbildung 5.1: Java-Code zur RTT-Bestimmung.

Im Datenteil der verschickten UDP-Pakete wird eine fortlaufende Seriennummer abgelegt, die bei zurückkommenden UDP-Paketen dazu benutzt wird, das zugehörige abgeschickte Paket zu identifizieren. Dadurch kann die Absendezeit des abgeschickten Pakets mit der Ankunftszeit des ankommenden Paketes verglichen und so die Echo-Laufzeit ermittelt werden. Des wei-

teren können im Datenteil der UDP-Pakete noch beliebige Inhalte untergebracht werden. Dies wird für den in Kapitel 4.4.3 erwähnten Hinweis auf den Hintergrund der Messungen genutzt.

`jping` kann beliebig viele Verbindungen gleichzeitig untersuchen. Dabei werden reihum an alle angegebenen Internet-Rechner Test-Pakete verschickt, erst das erste Paket an alle Rechner, dann das zweite an alle, und so fort. Um die angebbare maximale Paketrate für die Summe aller Messungen einzuhalten, wird zwischen dem Verschicken zweier Pakete eine entsprechend kalkulierte, konstante Zeit gewartet. Außerdem wird diese Zeitspanne so berechnet, dass zwei Test-Pakete an den gleichen Ziel-Rechner mindestens eine Sekunde Abstand haben.

5.2.2 Durchführung der Messungen

Die hier vorliegenden Ergebnisse resultieren aus Messungen, die vom 25. bis 29. November 1996 (fünf Werktage) durchgeführt wurden. Während dieser Zeit wurden zu jeder Stunde 10 Test-Pakete an jeden der ausgewählten Rechner gesendet. Es handelte sich dabei um 250 weltweit verteilte Rechner, die wie folgt ausgewählt wurden: aus den Logfiles des Proxy-Servers wurden die 500 am häufigsten angesprochenen WWW-Server ermittelt. Aus dieser Liste von diejenigen Adressen ausgefiltert, die keinen DNS-Eintrag (mehr) hatten. Dann wurde untersucht, an welchen dieser Adressen ein UDP-Echo-Port aktiv war. In ca. 40 % der Fälle war das nicht der Fall, weil der UDP-Echo-Dienst unterbunden wurde – vermutlich im Rahmen üblicher Sicherheitsmaßnahmen. Es blieben aber gut 250 Rechner übrig, die für die folgenden regelmäßigen Messungen in Frage kamen.

Für die im folgenden vorgestellten Ergebnisse wurden die ermittelten Paketlaufzeiten und Verlustraten über alle Werktage gleichmäßig gemittelt. Dabei wurde allerdings die Tageszeit unterschieden, da die Verbindungsqualität üblicherweise einem typischen täglichen Rhythmus schwankt.

5.2.3 Vergleich mit HTTP-Datentransferraten

Um einen Eindruck von der Bedeutung der gemessenen Werte zu bekommen, werden sie mit bekannten Leistungsparametern gängiger Dienste verglichen. Dazu wird im folgenden die Datentransferrate für das Übertragen von Dokumenten im WWW verwendet. Dies empfiehlt sich vor allem deshalb, weil 1) das WWW derzeit der am meisten benutzte Internet-Dienst ist und 2) weil auf dem Proxy-Server der Universität zu Köln eine große Zahl Verbindungen protokolliert wird. Die Daten sind also schon vorhanden, die Transaktionen wurden unabhängig von irgendwelchen Untersuchungen durchgeführt und die Notwendigkeit zu zusätzlichen Testmessungen entfällt.

Für die im folgenden verwendeten Werte wurden die Logfiles des Proxy-Servers für den 25. bis 29. November 1996 ausgewertet. Die Übertragungsra-

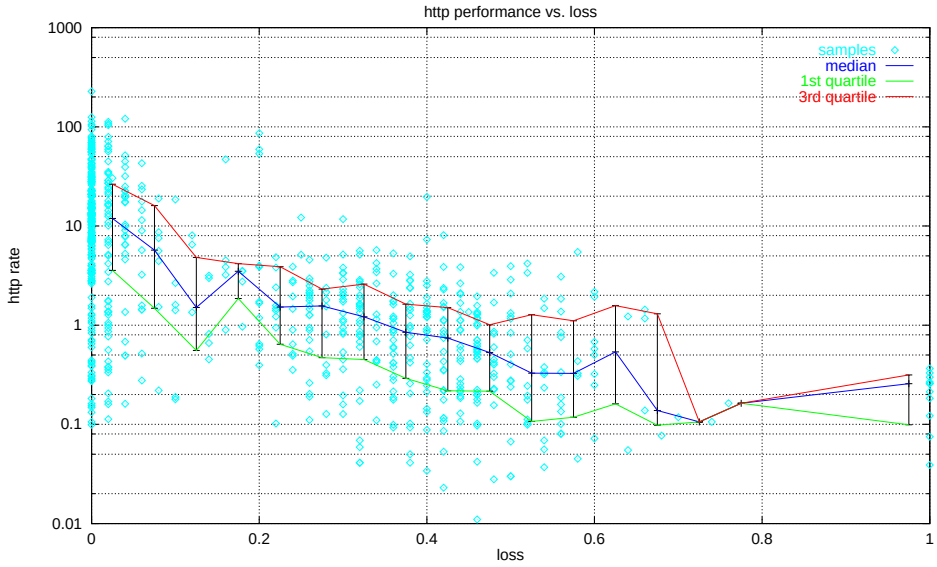

Abbildung 5.2: HTTP-Datentransferrate in Abhängigkeit der Verlustrate.

ten aller Transaktionen mit einem bestimmten WWW-Server wurden über die fünf genannten Tage gemittelt, allerdings wurde wiederum stundenweise die Tageszeit unterschieden.

Bild 5.2 zeigt für jede gemessene Kombination aus WWW-Server und Tageszeit die HTTP-Datentransferrate (in KB/s) auf der y-Achse in Abhängigkeit von der Paketverlustrate auf der x-Achse. Außerdem wird für gleichmäßige Intervalle auf der x-Achse der Median, das 25. und das 75. Perzentil aufgetragen. Man sieht deutlich, dass die Übertragungsrate mit steigender Paketverlustrate abnimmt.

Bild 5.3 zeigt in analoger Weise die Abhängigkeit der Datentransferrate von der Paketlaufzeit. Dabei wurden bei der Bestimmung der gemittelten Paketlaufzeit nur tatsächlich empfangene Pakete berücksichtigt. Verlorene Pakete wurden ignoriert. Man kann ahnen, dass einen längere Paketlaufzeit eine niedrigere Datentransferrate bewirkt, aber dieser Zusammenhang ist nicht in allen Bereichen deutlich.

Wesentlich deutlicher wird er, wenn man wie in Bild 5.4 auch Paketeverluste berücksichtigt. Hier wurde ein Paketverlust so gezählt, als wäre das Paket mit der maximalen Verzögerung zurückgekommen.

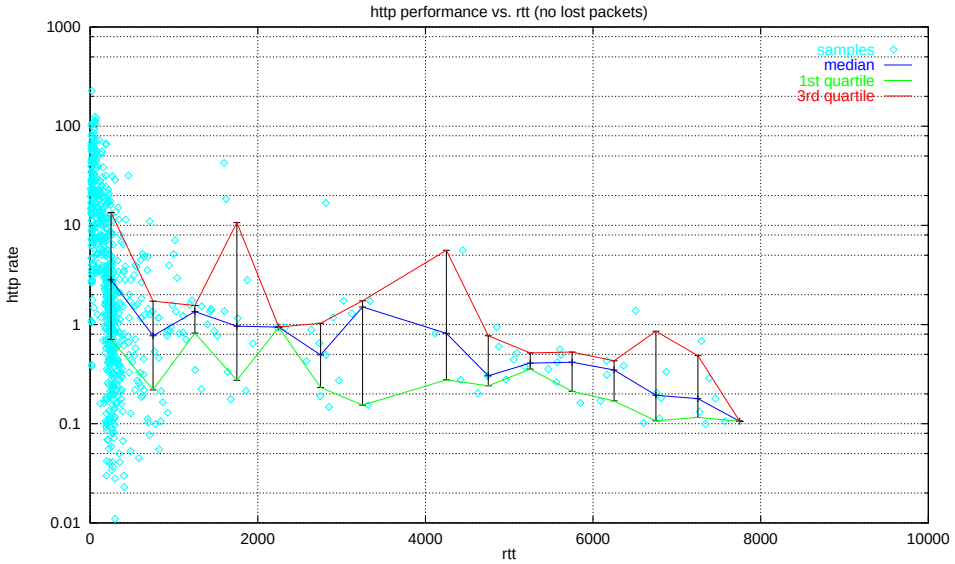

Abbildung 5.3: HTTP-Datentransferrate in Abhängigkeit der Paketlaufzeit, Paketverluste unberücksichtigt.

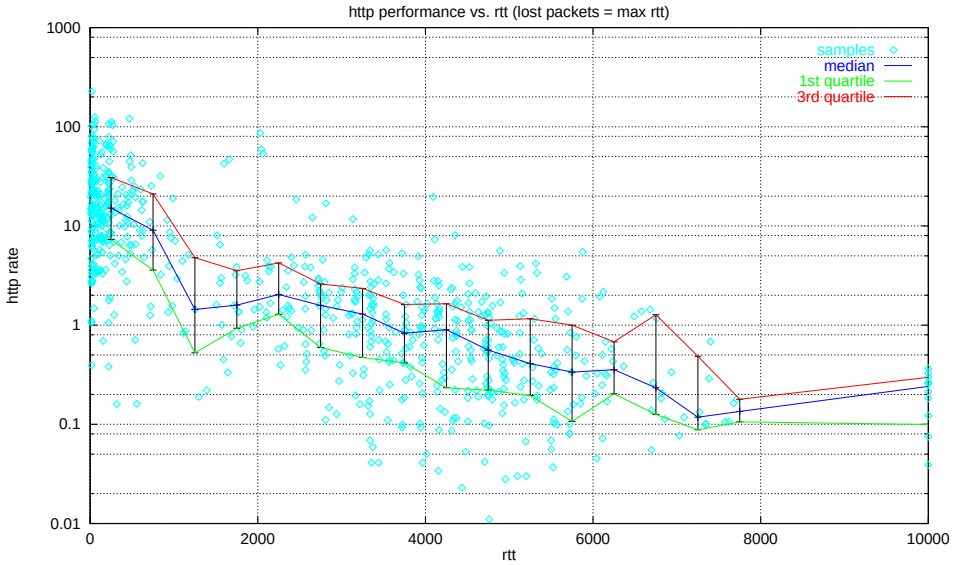

Abbildung 5.4: HTTP-Datentransferrate in Abhängigkeit der Paketlaufzeit, Paketverluste zählen wie maximale Paketlaufzeit.

5.3 Analyse des ersten Ansatzes

5.3.1 Genauigkeit der Zeitmessung

Problem Das Programm `jping` hat bei der Messung der Paketlaufzeit das Problem, nicht völlig exakt die Zeit ermitteln zu können, in der ein Paket tatsächlich den Rechner verlässt, beziehungsweise zu der es ankommt. Die einzige Möglichkeit ist die, unmittelbar nach dem Systemaufruf zum Absenden beziehungsweise Empfangen eines Pakets die Systemzeit abzufragen, siehe Abbildung 5.1. Diese hat allerdings in Java nur eine Auflösung von ganzen Millisekunden. Außerdem kommt für die Messungen üblicherweise kein Echtzeit-Betriebssystem zum Einsatz, so dass diese Art der Zeitmessung generell unexakt ist – zwischen dem Systemaufruf zum Absenden beziehungsweise Empfangen des Pakets und dem zum Abfragen der Systemzeit kann theoretisch beliebig viel Zeit vergehen.

Lösungsansatz Wenn der für die Messungen verwendete Rechner nicht überlastet ist, kann man jedoch erwarten, dass keine unerwarteten großen Wartezeiten auftreten und diese Art der Zeitmessung trotz kleinerer Ungenauigkeiten brauchbar ist.

Darüber hinaus bieten gängige Unix-Systeme einen Paketfilter-Mechanismus an, der es erlaubt, beliebige von einem bestimmten Netz-Interface verarbeitete Pakete direkt zu überwachen (wie zum Beispiel mit `tcpdump`). Der Kernel liefert bei dieser Schnittstelle eine Zeitangabe für jedes gefilterte Paket mit. Diese hat zum einen eine deutlich bessere Auslösung als die in Java abfragbare Systemzeit, und zwar bis herunter zu Nanosekunden. Zum anderen wird sie direkt vom Kernel erzeugt und hängt somit nicht von der Rechenzeit eines User-Level-Prozesses ab. In 5.5.1 wird die durch den Paketfilter-Mechanismus verbesserte Zeiterfassung durch Messungen im lokalen Netz untersucht und mit der alten Methode verglichen.

5.3.2 Nicht unterstützter UDP-Echo-Dienst

Problem Wie schon angedeutet antworten nicht alle Rechner im Internet auf Pakete, die an ihren UDP-Echo-Port gerichtet werden. Ein erheblicher Teil hat diesen Dienst, der zwar üblich aber nicht notwendig ist, gezielt abgestellt. Auch wenn man eine Liste von Gegenstellen im Internet zusammenstellt, bei denen der UDP-Port normal bedient wird, so kommt es doch immer wieder vor, dass im Laufe der Zeit bei immer mehr Rechnern dieser Liste der UDP-Echo-Port doch gesperrt wird. Die so abgewiesenen Test-Pakete gehen dann als „Verluste“ in die Statistik ein und verfälschen das Ergebnis, indem sie eine schlechtere Konnektivität anzeigen, obwohl nur an einigen Stellen eine restriktivere Administration eingeführt wurde.

Lösungsansatz Wenn ein Host im Internet ein Paket für den UDP-Echo-Port empfängt, diesen Dienst aber nicht anbietet, dann antwortet er mit einem ICMP-Paket vom Typ „port unreachable“ [77]. Wenn man diese ICMP-Pakete ebenso wie die im Normalfall zurückgeschickten UDP-Pakete als erfolgreiche Antwort wertet, kann man praktisch jeden Internet-Rechner mit korrekt implementiertem UDP-Protokoll für die Messungen verwenden.

Genauso wie die beschriebene genauere Zeitmessung lässt sich dies etwa mit Hilfe des Tools `tcpdump` erreichen. Damit ist es nämlich möglich, ICMP-Pakete genauso zu empfangen wie UDP-Pakete.

Dazu wurde das Tool `pingspy` entwickelt (siehe 5.4), das in der Lage ist, `tcpdump` in geeigneter Weise zu starten und seiner Ausgabe so zu verarbeiten, dass die entsprechende Art ICMP-Pakete ausgewertet werden kann.

5.3.3 Zuordnung der Echo-Pakete ohne Seriennummer

Problem Ein Problem ergibt sich leider aus der Tatsache, dass das Protokoll ICMP nur vorsieht, dass der 8 Bytes lange Header des UDP-Pakets im ICMP-Paket zurückgeschickt werden soll [66]. Die eigentlichen Daten des abgewiesenen UDP-Pakets können, müssen aber nicht mitgeschickt werden [77].

Falls die Daten des UDP-Pakets nicht mitgeschickt werden, fehlt die Seriennummer, die von `jping` benötigt wird, um die empfangenen Pakete den vorher abgesendeten zuzuordnen.

Lösungsansatz Man kann nun eine Zuordnung anhand der IP-Adresse der Gegenstelle vornehmen. D.h., man untersucht bei jedem ankommendem ICMP-Paket, ob noch eine Antwort auf ein Test-Paket zur IP-Adresse des zurückgeschickten UDP-Pakets aussteht, und ordnet es diesem Test-Paket zu.

5.3.4 Unterscheidung mehrerer ausstehender Echo-Pakete

Problem Damit gibt es allerdings Probleme, wenn mehrere Antworten für Pakete zu ein- und demselben Rechner ausstehen. Das ist beispielsweise dann der Fall, wenn weil die Paketlaufzeit größer ist als der Zeitabstand zwischen zwei Proben, oder wenn das vorhergehende Paket verloren gegangen war, etc. Dann ist die Zuordnung anhand der IP-Adresse nicht mehr eindeutig möglich.

Lösungsansatz Im 8 Bytes langen UDP-Header gibt es nun noch genau ein Feld, das sich vom Messprogramm aus zumindest teilweise beeinflussen lässt und in dem die zur Lösung des Problems notwendigen Informationen untergebracht werden können: die Portnummer des Absenders.

Man kann im Testprogramm mehrere verschiedene Verbindungsendpunkte[1] einrichten und für jedes Test-Paket zu einem bestimmten Host jeweils einen anderen Verbindungsendpunkt verwenden. Dann lässt sich mit Hilfe der Kombination aus IP-Adresse und Portnummer wieder eine eindeutige Zuordnung des ICMP-Pakets zum abgesendeten UDP-Paket herstellen. Zu diesem Zweck merkt sich `pingspy` auch die Quellportnummern der ausgesendeten Test-Pakete und zieht diese mit zur Zuordnung zurückkommender Echo-Pakete heran.

5.3.5 Multihomed-Rechner

Problem Weitere Probleme gibt es mit „multihomed“ Rechnern im Internet, also solchen, die mehrere Netzinterfaces mit unterschiedlichen IP-Adressen haben. Damit ist es nämlich möglich, dass UDP-Echo-Pakete den Rechner über ein bestimmtes Interface erreichen, dass sie aber auf einem anderen Interface beantwortet werden. Dann hat das zurückkommende ICMP-Paket eine IP-Adresse als Absender, die gar nicht mehr zugeordnet werden kann. Erste Untersuchungen zeigen, dass diese Art des asymmetrischen Routings durchaus recht häufig vorkommt (siehe auch 5.5.5).

Lösungsansatz Dieses Problem kann gelöst werden, indem nicht die Quell-Adresse des ICMP-Headers, sondern die Ziel-Adresse des im ICMP-Paket zurückgeschickten UDP-Headers ausgewertet wird. Der zurückgeschickte UDP-Header darf nämlich nicht verändert werden und enthält somit genau die IP-Adresse, die beim Versenden des Test-Pakets verwendet wurde. Auch dieses Vorgehen wurde in `pingspy` implementiert.

5.3.6 Verhindern ungewollter Synchronisation

Aus der klassischen Physik (und vielen anderen Fachbereichen) ist bekannt, dass lose gekoppelte schwingende Systeme häufig dazu neigen, sich früher oder später zu synchronisieren.

Dies gilt prinzipiell auch für periodische Vorgänge im Internet, da sich durch Zugriffsmechanismen, durch Warteschlangen in Routern etc. leicht eine kleine gegenseitige Beeinflussung, also eine „lose Koppelung“, ergeben kann. Floyd und Van Jacobson haben sich ausführlich mit dieser Problematik beschäftigt [24, 25]. Sie stellen eine Reihe von Beispielen vor, in denen Protokolle ohne die Befürchtung „zufälliger“ Synchronisation spezifiziert wurden, diese dann aber überraschend häufig doch eintritt und zu starker Netzbelastung und Performanceeinbußen führt. So wird die zyklische Vergrößerung und Verkleinerung der Window-Size genannt, wenn sich mehrere TCP-Verbindungen Ressourcen auf einem Router an einem gemeinsamen Flaschenhals teilen. Oder Client/Server-Modelle, in denen mehrere Clients

[1] sogenannte **Sockets** mit jeweils einer eindeutigen Portnummer

auf einen gemeinsamen Server warten und periodisch immer wieder die gleiche Anfrage abschicken. Und es wird auf Routing-Protokolle eingegangen, die in regelmäßigen Abständen Informationen austauschen und dann die Router für eine gewisse Zeit stark mit der Verarbeitung der neuen Routen belasten. Allen Beispielen gemein ist, dass periodisch irgendwelche Netzaktivitäten stattfinden und dass man nicht von vornherein Maßnahmen gegen unbeabsichtigte Synchronisation verschiedener Instanzen getroffen hat. Statt dessen war man nachher im praktischen Einsatz überrascht, dass so etwas überhaupt möglich ist.

Wenn man nun aktive Netz-Messungen in regelmäßigen Abständen durchführt, kann nicht ausgeschlossen werden, dass früher oder später ähnliche Probleme auftauchen. Außerdem haben Messungen mit genau festgelegter Periode den Nachteil, dass sich schon beim reinen Beobachten anderer periodischer Vorgänge Interferenzerscheinungen ergeben können, die die Ergebnisse verfälschen.

Als Konsequenz dieser Problematik fordert das Rahmenwerk der IPPM-Gruppe [59], dass jede Art regelmäßiger aktiver Messungen im Internet ein gewisses Maß an Zufall bei der Wahl der Messzeitpunkte verwenden soll.

Wenn T_n der Zeitpunkt der Aussendens des n-ten Testpakets an ein bestimmtes Ziel ist und $g_n = T_{n+1} - T_n$ der Abstand zwischen zwei Messungen, dann soll g_n keine Konstante, sondern ein Zufallswert sein. Am besten sollte g_n genau **Poisson-verteilt** sein. Dies ist genau dann der Fall, wenn g_n durch eine Zufallsgröße G mit der folgenden Verteilungsfunktion beschrieben wird:

$$G(t) = 1 - e^{-\lambda t} \tag{5.1}$$

mit λ: Senderate (in Paketen pro Sekunde)

Dann ist die Messung geschützt gegen die Erzeugung ungewollter Synchronisation, gegen interferenzartige Verfälschung der Messung periodischer Vorgänge und die Messzeitpunkte sind unvorhersagbar und somit gegen Manipulation geschützt.

Für eine Implementierung ist interessant, wie sich eine negativ-exponentiell verteilte Wartezeit g_n aus einer gleichförmig verteilten Pseudo-Zufallszahl u_n bilden lässt:

$$\begin{aligned} & u_n = 1 - e^{-\lambda g_n} \\ \Leftrightarrow \quad & g_n = -\frac{\ln(1-u_n)}{\lambda} \end{aligned} \tag{5.2}$$

Leider ist es praktisch unmöglich, den Zeitpunkt des Aussendens eines Pakets genau festzulegen. Daher implementiert `jping` die folgende Näherung: Nach dem Aussenden eines Testpakets wird eine zufällig ausgewählte, negativ-exponentiell verteilte Zeitspanne g_n gewartet und dann das nächste Paket zum gleichen Ziel gesendet. Der Unterschied zum optimalen Verfahren besteht dann nur in einer mehr oder weniger konstanten Größe c_n, die

zu jeder Wartezeit hinzuaddiert wird. Die tatsächlichen Sendezeitpunkte T'_n ergeben sich dann wie folgt:

$$T'_{n+1} = T'_n + g_n + c_n \tag{5.3}$$

RFC 2330 [59] beschreibt noch zwei andere mögliche Implementierungen, die dem theoretischen Ideal unter Umständen näherkommen. Es sagt aber auch, dass der Aufwand dafür gegebenenfalls erheblich größer ist, dass kein Anhaltspunkt dafür besteht, dass die erreichbare Genauigkeit der Messzeitpunkte für die Ergebnisse überhaupt relevant ist und dass die aufwändigeren Implementierungen im Falle gleichzeitig stattfindender paralleler Messungen unter Umständen sogar schlechter sind. Da `jping` aber tatsächlich mehrere Messungen quasi parallel ausführt, wird von diesen Alternativen Abstand genommen. Solange die mittlere Wartezeit $\bar{g}$ deutlich größer ist als die für die Messung benötigte Rechenzeit $\bar{c}$, bleiben die gewünschten Eigenschaften im wesentlichen erhalten. So verlangen Floyd und Van Jacobson beispielsweise nur, dass $\bar{g}$ doppelt so groß sein soll wie $\bar{c}$, um ungewollter Synchronisation vorzubeugen [24, 25].

5.4 Verbesserter Ansatz: Implementierung von `pingspy`

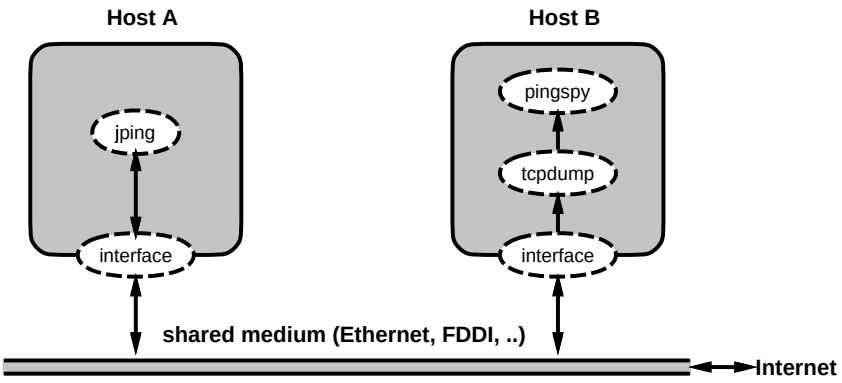

Abbildung 5.5: Struktur von `pingspy` beim Einsatz eines geteilten Netzmediums.

Implementiert wurde das verfeinerte Verfahren in dem Perl-Programm `pingspy.pl`, das `tcpdump` mit einer geeigneten Filterbedingung startet und parallel zum bekannten Java-Programm `jping` läuft. Das Java-Programm generiert wie gehabt die Test-Pakete und das Perl-Programm registriert die Test-Pakete, alle UDP- und ICMP-Antworten und erzeugt am Ende geeignete Logfiles.

Bei dieser Aufgabenteilung ist es möglich, beide Programm auf verschiedenen Rechnern laufen zu lassen, wenn diese über ein gemeinsames shared medium untereinander und mit dem Internet verbunden sind (wie zum Beispiel konventionelles Ethernet oder FDDI). Dies ergibt den Aufbau, wie er in Abbildung 5.5 gezeigt wird. Er hat gegenüber der anderen Variante den Vorteil, dass die Zeit, zu der die ausgesendeten Pakete tatsächlich den ersten Rechner verlassen, am genauesten bestimmt werden kann.

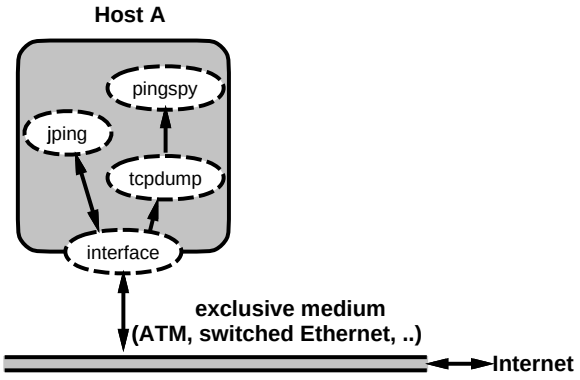

Abbildung 5.6: Struktur von `pingspy` beim Einsatz eines exklusiven Mediums.

Wenn kein zusätzlicher Rechner zum Mithören der Pakete zu Verfügung steht oder wenn ein exklusiver Netzanschluss (etwa ATM oder geswitchtes Ethernet) zu Einsatz kommt, dann können beide Programme auch auf dem gleichen Rechner laufen, wie in Abbildung 5.6 dargestellt.

Das Java-Programm `jping` wurde überarbeitet, um den Anforderungen aus 5.3.4 und 5.3.6 gerecht zu werden.

So werden nun bis zu 50 unterschiedliche lokale Portnummern verwendet (entsprechend 50 geöffneten Sockets), um eine Zuordnung von zurückkehrenden UDP-Headern auch ohne deren Datenteil zu ermöglichen (siehe 5.3.4).

Des weiteren werden mehrere parallel laufende Threads eingerichtet, von denen einer zu einer Zeit nur zum Absenden der Testpakete an *ein* bestimmtes Ziel zuständig ist. Dabei wird die Wartezeit gemäß 5.3.6 berechnet und möglichst genau eingehalten.

Ein weiterer Thread dient zum Empfangen ankommender Pakete und läuft mit erhöhter Priorität, damit er die Bestimmung der Ankunftszeit möglichst genau vornehmen kann. Schließlich läuft ein Thread mit niedriger Priorität im Hintergrund, um laufend die Auswertung der angekommenen Pakete vorzunehmen, ohne die anderen Threads in den kritischen Momenten zu beeinträchtigen.

5.5 Analyse des verbesserten Ansatzes

5.5.1 Genauigkeit der Zeitmessung

Zunächst sollte untersucht werden, wie sehr die Genauigkeit der Zeitmessung durch die Verwendung eines Paketfilter nun tatsächlich verbessert werden konnte, beziehungsweise wie stark die einfache Zeitmessung des Java-Programms die tatsächlichen Werte verfälscht.

Die Auflösung der Java-Systemzeit kann man für Abweichungen von maximal einer Millisekunde verantwortlich machen. Darüber hinaus liegt eine Fehlerquelle in der Programmlaufzeit vom Systemaufruf zum Absenden beziehungsweise Empfangen des Pakets bis zur Abfrage des Systemzeit. Solange man kein Echtzeit-Betriebssystem verwendet, kann diese Zeit nicht a priori abgeschätzt werden. Wenn das Test-Programm auf einem nur wenig belasteten Rechner läuft, kann man immerhin erwarten, dass diese Zeit nicht übermäßig lang sein wird und auf modernen Computern den Bereich weniger Millisekunden normalerweise nicht übersteigt.

Paketlaufzeiten im WAN-Bereich liegen hingegen häufig im Bereich mehrerer hundert Millisekunden, so dass Messungen im WAN alleine keine guten Rückschlüsse auf die Genauigkeit der Zeitmessung erlauben. Deshalb wurden speziell zu diesem Zweck einige Messungen im lokalen Netz durchgeführt.

Name	Typ / Betriebssystem	Netzanbindung	Last
axp1	DEC-7000 (Alpha) Digital UNIX V3.2	FDDI, ein Router	hoch (Batch-Betrieb)
fs2	SPARCstation 20 Solaris 2.4	FDDI, ein Router	mittel (Fileserver)
pelican	DEC (Alpha) Digital UNIX V4.0	direkt (Ethernet)	keine
rs1	IBM RS/6000 AIX 3.2.5	FDDI, ein Router	teilw. sehr hoch (div. Server)
starfire	SUN Ultra E10000 Solaris 2.5.1	direkt (ATM-622)	mittel (Batch-Betrieb)
trex	SUN Ultra 2 Solaris 2.5.1	direkt (ATM-155)	niedrig (WWW-Cache)

Tabelle 5.1: Im lokalen Test verwendete Ziel-Systeme.

Als Test-Rechner wurde eine SUN-Workstation vom Typ Ultra 2 verwendet, die mit zwei Prozessoren und einem 622-Mbps-ATM-Netzinterface ausgestattet war. Zum Zeitpunkt der Untersuchung stellte diese Ausstattung den Stand der Technik dar und ermöglichte eine optimale Messung. Als Ziel der Messungen wurden verschiedene Workstations und Server mit unterschiedlicher Architektur und Betriebssystemen ausgewählt. Tabelle 5.1

zeigt die für die lokale Messungen verwendeten Systeme.

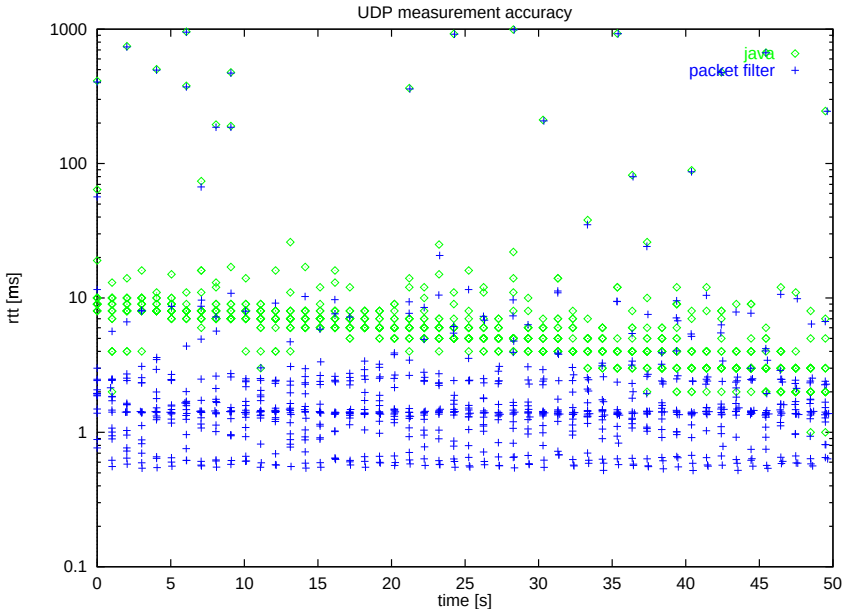

Abbildung 5.7: Genauigkeit der RTT-Messung im Java-Programm und mittels Paketfilter im lokalen Test.

Abbildung 5.7 zeigt die Ergebnisse einer Messung mit dem UDP-Echo-Port der Zielsystemen. Von 850 ausgesendeten Test-Paketen kamen 848 zurück, was einem Paketverlust von 0.2 % entspricht. Die Test-Pakete wurden in Serien zu je 50 Stück ausgesendet, wobei die Pakete innerhalb einer Serie jeweils im Abstand von einer Sekunde gesendet wurden. Die x-Achse zeigt für jedes Paket die Zeit des Absendens gemessen seit Beginn der Serie und die y-Achse zeigt die Zeit, die das Paket bis zu seiner Rückkehr unterwegs war (Round-Trip-Time, RTT). Für jedes Paket wird sowohl die vom Java-Programm, als auch die vom Paketfilter gemessene Zeit eingetragen. Man kann schon erkennen, dass die vom Java-Programm gemessenen Werte i.d.R. deutlich höher liegen.

Um diese Abweichung genauer zu quantifizieren, zeigt Abbildung 5.8 die relative Verteilungsdichte der gemessenen RTT-Werte. Da die verschiedenen Zielsysteme jeweils eigene charakteristische Maxima haben, ergeben sich in der Summe Kurven mit mehreren Maxima.

Aussagekräftige Zahlen lassen sich am besten aus der Verteilungsfunktion gewinnen, wie sie in Abbildung 5.9 dargestellt werden. Außerdem wird für jede Kurve der Median eingetragen. Während die (grobe) Messung des Java-Programms einen Median von sechs Sekunden ergibt, liefert die genauere Messung mittels Paketfilter einen Median von 1.42. Im Mittel liefert das Java-Programm also Werte, die etwa fünf Sekunden zu hoch liegen.

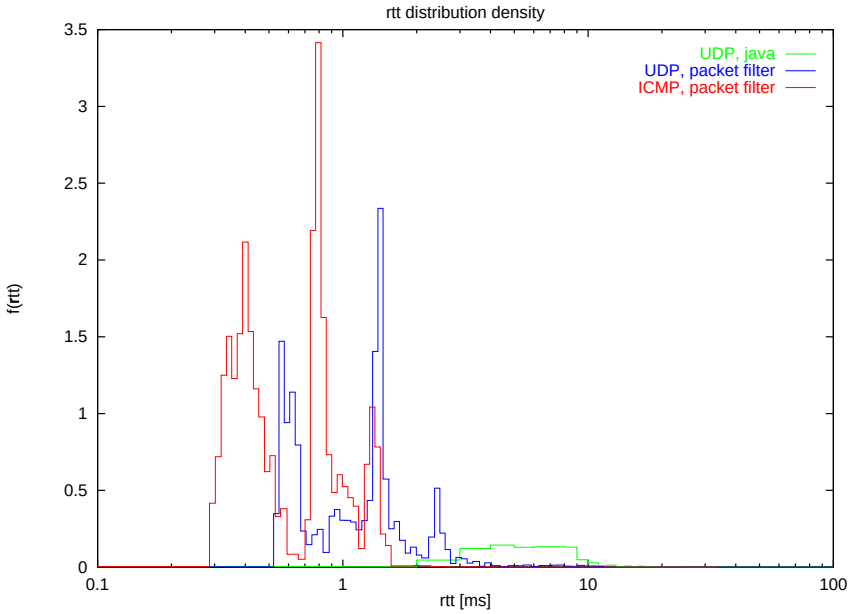

Abbildung 5.8: Verteilungsdichte der gemessenen RTT-Werte.

5.5.2 Einfluss der Bearbeitungszeit

Des weiteren stellt sich die Frage, inwieweit bei der Beantwortung von UDP-Echo-Paketen im Fall von UDP- und ICMP-Antworten unterschiedliche Bearbeitungszeiten auftreten und ob sie eine Rolle für die Bestimmung der Paketlaufzeit spielen. Da ICMP-Antworten typischerweise direkt vom Kernel generiert werden, eine normale Bedienung des UDP-Echo-Ports aber im `inetd`-Prozess stattfindet, kann man erwarten, dass ICMP-Antworten etwas schneller sind. Es entfällt nämlich der Aufwand für die Kommunikation mit dem `inetd`-Prozess und es werden mindestens zwei Kontextwechsel vermieden.

Um dies näher zu untersuchen, wurden die gleichen Messungen wie in Kapitel 5.5.1 auch mit anderen UDP-Ports durchgeführt, die von den Ziel-Rechnern *nicht* bedient werden. Das heißt, es wurden in jedem Fall nur ICMP-Antworten erzeugt. Bei dieser Messungen wurden 900 Test-Pakete verschickt und 881 ICMP-Antworten gezählt, was einen Verlustrate von etwa 2 % entspricht (siehe auch Kapitel 5.5.6).

Die gemessenen Verteilungen sind in den Abbildungen 5.8 und 5.9 bereits eingetragen. Sie zeigen deutlich ein noch etwas schnelleres Antwortverhalten. Der Median der ICMP-Antworten liegt bei 0.80 (statt 1.42 für UDP-Antworten). Dieser Unterschied von weniger als einer Millisekunde ist allerdings vernachlässigbar klein im Vergleich zu den Paketlaufzeiten, die typischerweise bei WAN-Verbindungen auftreten.

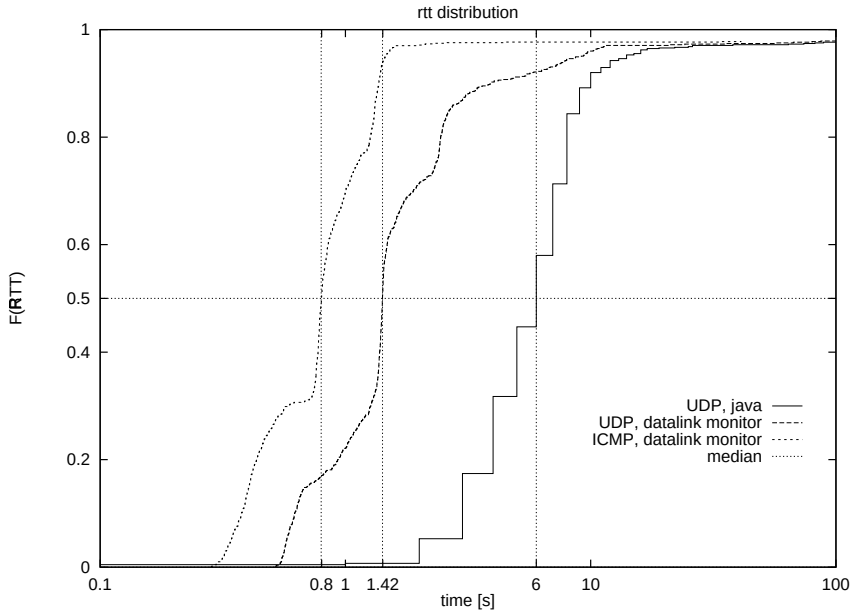

Abbildung 5.9: Median und Verteilung der gemessenen RTT-Werte.

5.5.3 First-Packet-Effekte

Besonderes Augenmerk gebührt jeweils dem ersten Paket in einer Serie. Im Internet gibt es viele Stellen, an denen die Bearbeitung des ersten Pakets einer Serie potenziell aufwändiger ist, als für zeitlich kurz darauffolgende „ähnliche“ Pakete.

- Das kann beispielsweise eine Firewall sein, die die Absender-Adresse aller Pakete überprüft und dafür eine DNS-Anfrage tätigen muss. Diese ist beim ersten mal deutlich aufwändiger, als wenn die Antwort schon in ihrem Cache liegt.

- Praktisch alle gängigen Router an zentralen Stellen verwenden heutzutage irgendeine Art Caching, um Routing-Entscheidungen effizienter zu machen. Diese Entscheidungen können je nach Konfiguration des Router recht aufwändig sein, müssen aber dank Caching nur für das erste Paket einer Serie neu getroffen werden.

- Auch für Dienste wie den UDP-Echo-Dienst kann es Initialisierungs- und Caching-Vorgänge geben, die zu einer langsameren Bearbeitung speziell des ersten Pakets führen. Zum Beispiel ist es denkbar, dass der Code für den UDP-Echo-Dienst auf einem stark belasteten Unix-System nicht immer im realem Hauptspeicher gehalten wird, sondern per demand paging auf Festplatte ausgelagert wurde. Nur das erste

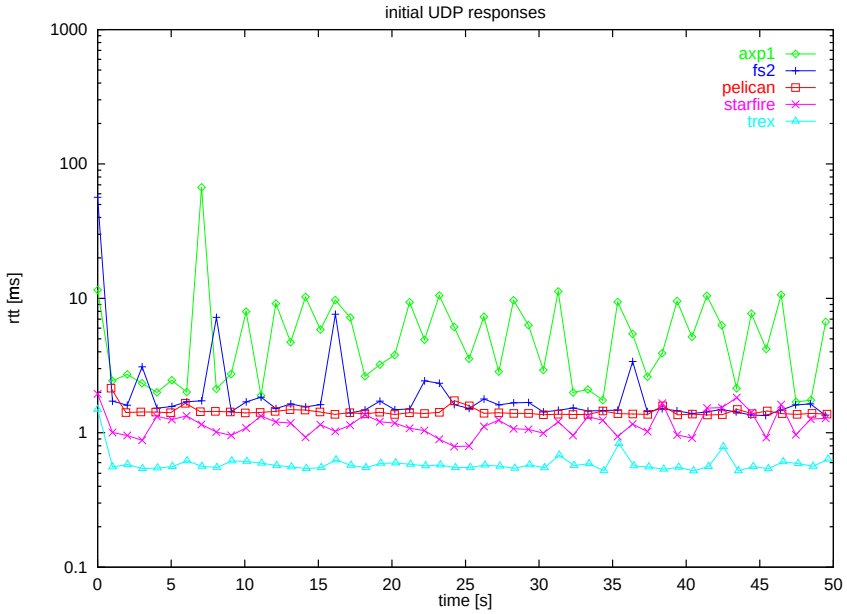

Abbildung 5.10: Verzögerung des ersten Pakets bei Verwendung des UDP-Echo-Ports.

Paket aus einer Serie muss dann warten, bis die nötigen Pages geladen wurden.

Abbildung 5.10 zeigt den Verlauf der ersten Serie für die verschiedenen Systeme. In praktisch allen Fällen braucht das erste Paket deutlich länger als die folgenden. Der Unterschied liegt je nach System und Last des Systems im Bereich von unter einer bis über 50 Millisekunden.

Abbildung 5.11 zeigt das gleiche für die erste Serie der Messung mit ICMP-Antworten. Hier lässt sich kein „first-packet“–Effekt beobachten. Auch dies lässt sich damit erklären, dass die ICMP-Antworten im Kernel erzeugt werden und damit nicht dem demand paging unterliegen wie der `inetd`-Prozess.

Eine der im Rahmen der beschriebenen Tests durchgeführten Messungen fiel vollkommen aus dem Rahmen. Sie ist in Abbildung 5.12 dargestellt: Die ersten Pakete kommen erst nach über 27 Sekunden zurück! Diese ungewöhnlich lange Antwortzeit normalisiert sich im Laufe von ca. 30 Sekunden. Die Maschine, zu der in diesem Fall gemessen wurde, ist bekannt dafür, dass sie häufig sehr hohe Lastspitzen hat. Offenbar führte eine solche Lastspitze dazu, dass der Prozess `inetd` für über 27 Sekunden nicht laufen konnte und dann eine Reihe angestauter Pakete „am Stück“ abgearbeitet hat.

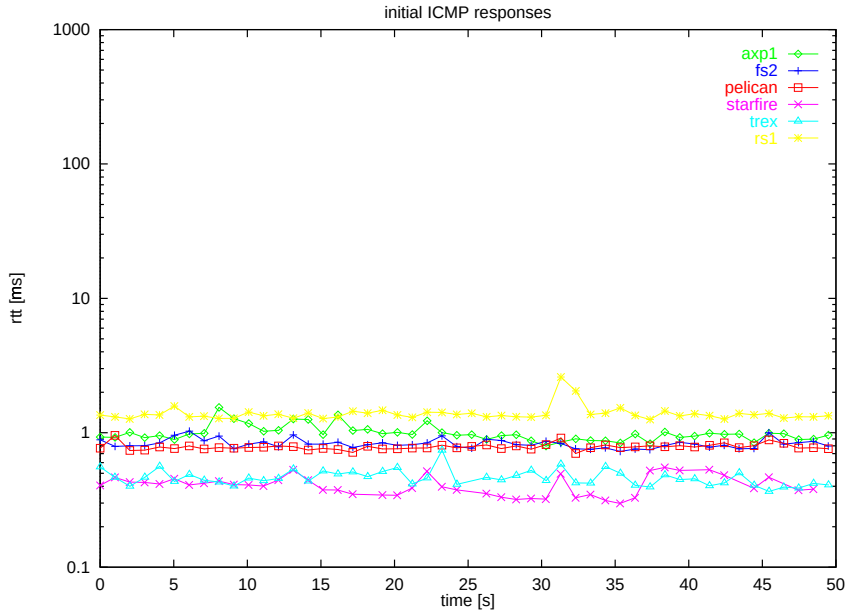

Abbildung 5.11: Verzögerung des ersten Pakets bei ICMP-Antworten.

5.5.4 First-Packet-Effekte im Weitbereichsverkehr

Auch im Weitbereichsverkehr gibt es messbare First-Packet-Effekte, die allerdings etwas andere Hauptursachen haben als die in lokalen Netzen. Während die in Kapitel 5.5.3 gemessenen Erstpaketverzögerungen hauptsächlich auf Mechanismen im antwortenden System zurückgeführt werden konnten, treten bei den längeren Paketlaufzeiten im Weitbereichsverkehr andere Effekte in den Vordergrund. Anders als in lokalen Netzen betreffen weitere Verbindungen im Internet meist viele zwischengeschaltete Router, so dass sich deren Erstpaketverzögerungen addieren. Außerdem gibt es unter den Routern im Internet eine große Vielfalt unterschiedlicher Produkte, die teilweise sehr starke First-Packet-Effekte aufweisen.

So kommt es, dass schon dann eine deutliche Erstpaketverzögerung sichtbar wird, wenn über viele untersuchte Verbindungen gemittelt wird und die normale durchschnittliche Paketlaufzeit recht hoch ist. Abbildung 5.13 zeigt über ca. 1000 Verbindungen gemittelt den Verlauf der Paketlaufzeit[2]. Dabei werden die erhaltenen UDP-Echo- und ICMP-Antworten getrennt voneinander in zwei Kurven dargestellt.

Um beim Messen der Paketlaufzeiten besser reproduzierbare und vergleichbare Ergebnisse zu erhalten, sollte man vorhersehbare First-Packet-Effekte möglichst ausschalten. Dies kann geschehen, indem man die ersten

[2]Die hier zugrunde liegenden Daten wurden vom 20. bis 31.8.1997 an der Universität zu Köln gemessen.

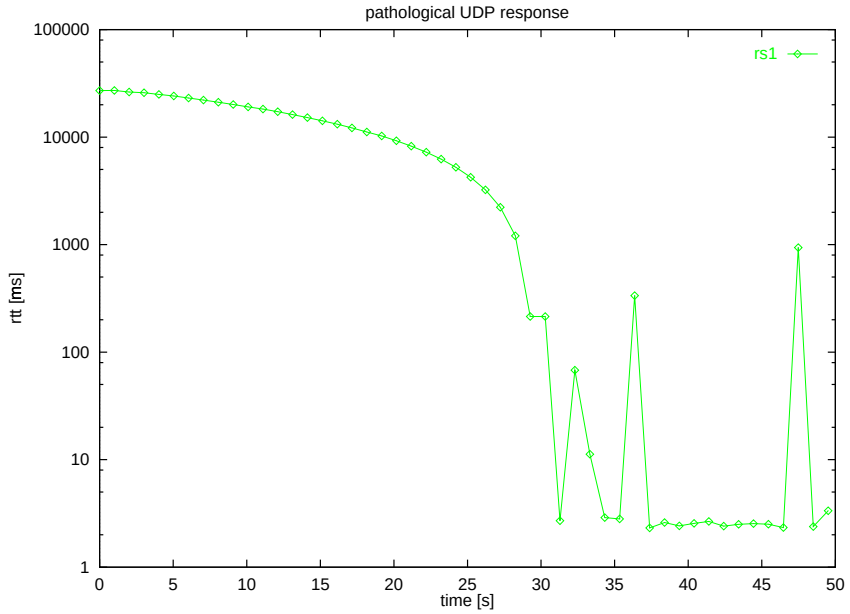

Abbildung 5.12: Überlange UDP-Antwortzeiten bei überlastetem Server.

k Pakete einer Serie verwirft und nur die Messergebnisse der folgenden Pakete berücksichtigt. Solange die Abstände zwischen den einzelnen Pakete nicht zu groß werden, stehen die Routing-Entscheidungen nach dem ersten Paket bei allen betroffenen Routern im Cache und haben dann eine geringere Auswirkung auf die Paketlaufzeit der folgenden Pakete. Die Frage ist, wie viele Pakete am Anfang jeder Serie ignoriert werden müssen, damit die gewünschte Stabilität eintritt.

Um dies genauer untersuchen zu können, wird die **relative Erstpaketverzögerung** RID einer Serie als Verhältnis der Erstpaketlaufzeit zur mittleren Laufzeit der restlichen Pakete definiert, wobei die k ersten Pakete ganz ignoriert werden:

$$RID = \frac{\tau_{k+1}}{\frac{1}{n-k-1} \sum_{i=k+2}^{n} \tau_i} \tag{5.4}$$

mit
- RID : relative Erstpaketverzögerung (Relative Initial Delay),
- τ_i : Laufzeit des i-ten Pakets einer Serie,
- n : Anzahl der Pakete einer Serie,
- k : Anzahl der zu ignorierenden Pakete.

Wird dieser Wert nun für jede Messserie berechnet, so sollte ein Wert von $RID = 1$ für den Fall stehen, dass die Paketlaufzeit für alle Pakete gleichverteilt ist. Liegt der Wert deutlich über 1, so deutet dies auf eine starke zusätzliche Verzögerung des ersten Pakets hin.

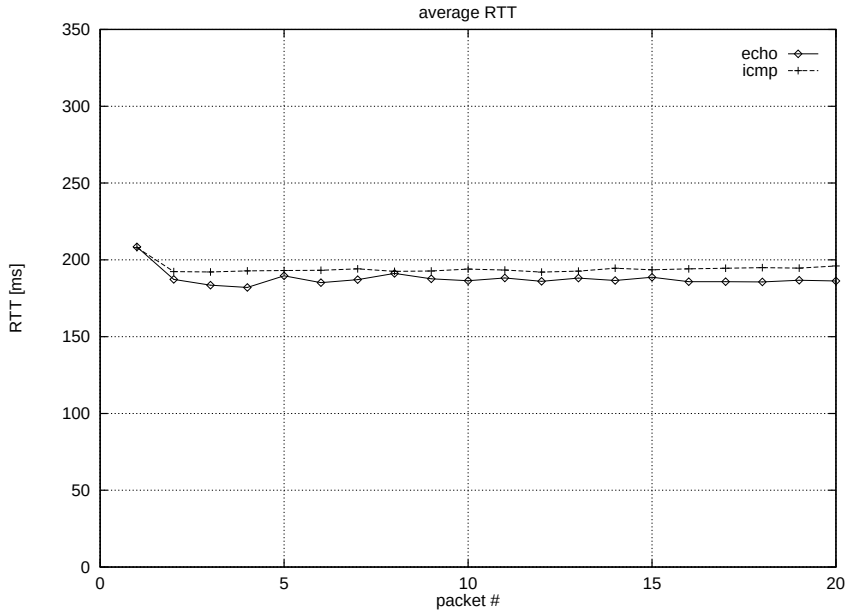

Abbildung 5.13: Paketverzögerungen im Weitbereichsverkehr.

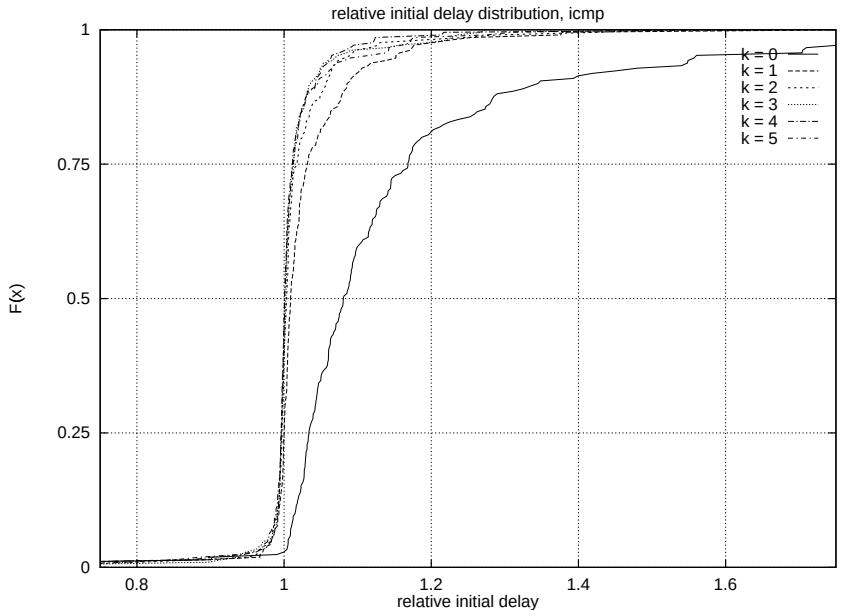

Abbildung 5.14: Relative Erstpaketverzögerung von ICMP-Antworten bei Nichtberücksichtigung der ersten k Pakete einer Serie.

Abbildung 5.14 zeigt die Verteilungsfunktion der relative Erstpaketverzögerung für verschiedene Werte von k. Man kann sehen, dass der Median für $k = 0$ deutlich über 1 liegt und sich für größere Werte von k an 1 annähert. Ein Wert von $k = 1$, also die Verwendung eines einzelnen Pakets zur Initialisierung der Internet-Verbindung, reicht allerdings noch nicht aus, um First-Packet-Effekte gut auszuschalten.

Statt nun generell einen höheren Wert für k zu wählen, kann man k auch für jede Serie so anpassen, dass alle Pakete, die eine Zeit ΔT nach dem ersten Paket abgeschickt werden, ignoriert werden. Anders gesagt: man wählt k so, dass gilt

$$\begin{array}{llll} T_i < T_1 + \Delta T & \text{für alle} & i < k & \text{und} \\ T_i \geq T_1 + \Delta T & \text{für alle} & i \geq k & \end{array} \tag{5.5}$$

mit
T_i : Sendezeitpunkt des i-ten Pakets einer Serie,
ΔT : Dauer der zu ignorierenden Anfangsphase einer Serie.

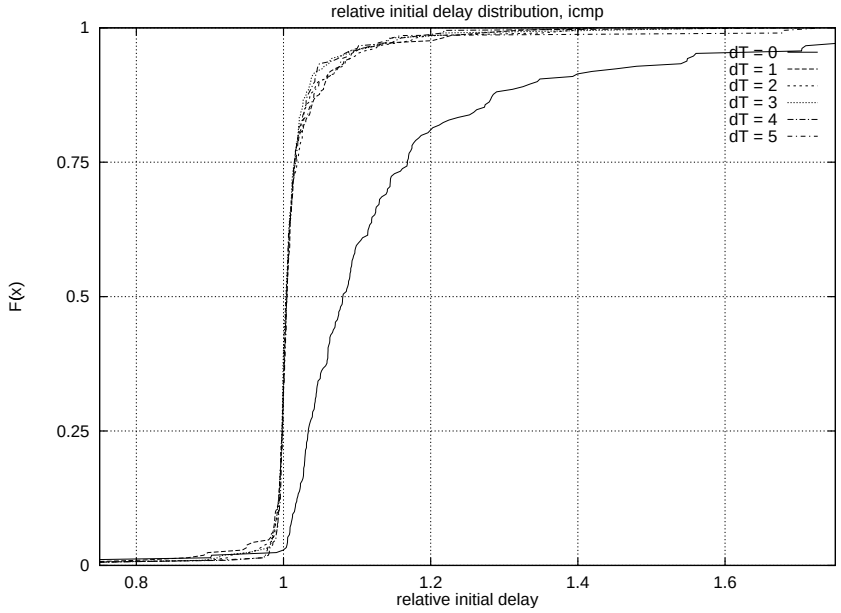

Abbildung 5.15: Relative Erstpaketverzögerung von ICMP-Antworten bei Nichtberücksichtigung der ersten Sekunden einer Serie.

Abbildung 5.15 zeigt wieder die Verteilung der relativen Erstpaketverzögerung, allerdings bei vorgegebenem ΔT, das im Bereich weniger Sekunden variiert wird. Man kann sehen, dass bereits das Überspringen der ersten Sekunde einer Serie vollkommen ausreicht, um First-Packet-Effekte praktisch ganz auszuschließen.

5.5.5 Antwortverhalten von Hosts im Internet

Die Hosts im Internet, die von `jping` angesprochen werden, können im wesentlichen auf fünf verschiedene Arten antworten, die im folgen unterschieden werden:

echo Das UDP-Paket an den Echo-Port des Zielsystems wird mit einem UDP-Paket beantwortet. Dies ist der Normalfall; der UDP-Echo-Dienst ist verfügbar und wurde nicht speziell abgestellt.

Für die gewünschten Messungen ist diese Reaktion gut geeignet, wenngleich unbekannte Verzögerungen durch Kontextwechsel und demand paging auf der Zielmaschine die Messung der RTT etwas verfälschen können (Kapitel 5.5.2, 5.5.3).

ICMP (no data) Der UDP-Echo-Port wird vom Zielsystem zwar nicht bedient, aber es antwortet mit einer korrekten ICMP-Fehlermeldung vom Typ „port unreachable“ [77]. Dabei schickt der Host entsprechend [66] mit der Fehlermeldung nur den Header der ursprünglichen UDP-Pakets, nicht aber den Applikationsdaten-Teil zurück. Auch Antworten dieses Typs lassen sich von `jping` auswerten (5.3.4, 5.4) und sind somit für die Messungen geeignet. Wie angedeutet ist die erreichbare Messgenauigkeit für sogar etwas besser als bei einfachen UDP-Echo-Antworten.

ICMP (data) Auch hier antwortet das Zielsystem mit einer ICMP-Fehlermeldung vom Typ „port unreachable“, schickt hier allerdings noch einige Applikations-Daten aus dem UDP-Paket mit.

Dies ist an und für sich durchaus wünschenswert, da es das Zuordnen der Antworten zu den Test-Paketen erleichtert (siehe Kapitel 5.3.3) und sonst alle Vorteile der ICMP-Antworten aufweist (siehe oben).

Leider zeigt sich in der Praxis jedoch, dass die meisten Systeme, die auf diese Art antworten, andere Nachteile haben, die ihre Brauchbarkeit für die gewünschten Messungen deutlich einschränken. Dies wird in Kapitel 5.5.6 näher erörtert.

error Schließlich gibt es eine Reihe weiterer ICMP-Fehlermeldungen, die auf Probleme der Vermittlung des IP-Pakets hindeuten. Sie werden nicht vom Ziel-System, sondern von einem auf dem Weg dorthin liegenden Router erzeugt und lassen sich daher nicht sinnvoll für die Messung auswerten. In der Praxis kommen folgende ICMP-Fehlermeldungen am häufigsten vor:

communication administratively prohibited Das Zielsystem befindet sich in einer Firewall-Umgebung und hat keinen freien Zugang zum Internet. Ein Router mit Paketfilter lässt vielmehr nur

sehr gezielt ausgewählte Pakete zum Zielsystem durch, beispielsweise solche, die zur Erbringung eines bestimmten Dienstes dienen. Alle anderen IP-Pakete werden mit einer Fehlermeldung dieser Art abgewiesen.

host unreachable Es existiert keine Route zum Zielsystem. Der erste Router, der dies feststellt, erzeugt diese Fehlermeldung. Dies tritt typischerweise nur vorübergehend auf, etwa wenn eine nichtredundante Verbindung ausfällt.

time-to-live exceeded in transit Das TTL-Feld ist vor Erreichen des Zielsystems abgelaufen. Das passiert normalerweise nur, wenn es wegen Routing-Fehlern zur Schleifenbildung kommt und Paket im Kreis herumgeschickt werden. Solche Situationen kommen gelegentlich vor, sind typischerweise aber nur von kurzer Dauer.

keine Antwort Schließlich kann es auch passieren, dass auf ein Test-Paket gar keine Antwort erfolgt. Das kann zu einem an einem Paketverluste liegen, entweder des Testpakets auf dem Hinweg oder der Antwort auf dem Rückweg. Zum anderen kommt es auch vor, dass ein Host generell nicht antwortet, etwas weil er außer Betrieb genommen wurde und gar nicht mehr im Internet existiert, oder weil er durch eine sehr restriktive Firewall abgeschirmt wird, die die Testpakete kommentarlos verwirft.

Bei der Ermittlung der Paketverlustrate stellt sich nun die Frage, was mit den nicht verwertbaren und den fehlenden Antworten zu tun ist. Wertet man alle nicht brauchbaren Antworten als Paketverlust, so tut man dem zugrundeliegenden Netz unrecht. Durch diejenigen Hosts, die sowieso keine brauchbaren Antworten liefern, erhält man stark verfälschte Mittelwerte für die Paketverlustrate. Die einzige Lösung ist es daher, die Hosts auf ihr typisches Antwortverhalten hin zu untersuchen und zu klassifizieren und dann für die Berechnung von netzweiten Mittelwerten nur die Hosts aus brauchbaren Klassen heranzuziehen.

Abbildung 5.16 zeigt für jeden von 1000 untersuchten Hosts den Anteil der beobachteten Antworten auf die Testpakete. Dabei gibt die unterste Kurve den Anteil der **echo**–Antworten an, die nächste gibt darüber den Anteil der **icmp (no data)**–Antworten an, etc. Die Hosts sind auf der x-Achse so sortiert, dass Hosts mit überwiegend gleichem Antwortverhalten benachbart und mit zunehmenden Paketverlustrate dargestellt werden. Dadurch wird deutlich, wie sich die Hosts in verschiedene Klassen aufteilen lassen. Tabelle 5.2 zeigt die Anteile der Hosts an den verschiedenen Klassen.

Innerhalb der brauchbaren Klassen **echo** und **icmp (no data)** lässt sich in Abbildung 5.16 eine charakteristische Verteilung der Paketverlustrate erkennen. Daraus, dass der Verlauf in beiden Klassen gleichartig ist, kann man schon sehen, dass die so gemessene Paketverlustrate eine Eigenschaft des Netzes und unabhängig von der Art der Antwort ist.

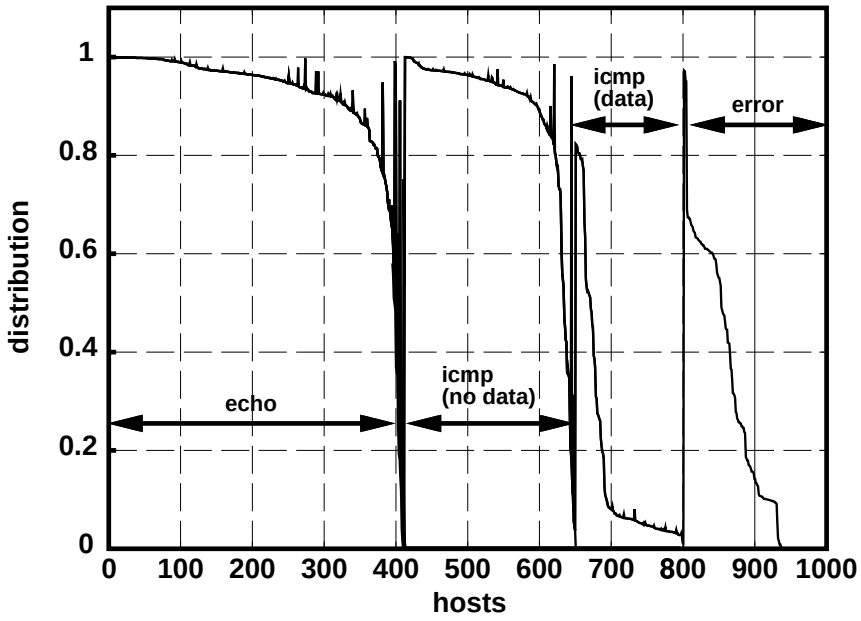

Abbildung 5.16: Verteilung der möglichen Antworten.

In den beiden Klassen der unbrauchbaren Antworten, **icmp (data)** und **error** sieht der Kurvenverlauf ganz anders aus. Insbesondere fällt die in weiten Bereichen scheinbar massive Verlustrate der Klasse **icmp (data)** auf. Dies wird in Kapitel 5.5.6 erklärt.

Es ist auch ein guter Grund, tatsächlich weiterhin den UDP-Echo-Port für die Testpakete zu verwenden. Nach den Erkenntnissen aus Kapitel 5.5.2 könnte man sonst zunächst folgern, dass die Verwendung eines unbekannten Ziel-Ports sinnvoller ist, weil dann immer ICMP-Antworten erzeugt werden und die Messgenauigkeit der Paketlaufzeit steigt. Allerdings würde damit auch der Anteil der Hosts steigen, die sich nicht zur Bestimmung der Paketverlustrate eignen. Um eine möglichst große Datenbasis zu haben, wird

Antwort-Klasse	Anzahl Hosts		%	
	pro Klasse	summiert	pro Klasse	summiert
echo	412	412	41.4	41.4
icmp (no data)	239	651	24.0	65.5
icmp (data)	151	802	15.2	80.6
error	150	952	15.1	95.8
(keine Antwort)	42	994	4.2	100.0

Tabelle 5.2: Anteile der verschiedenen Hostklassen.

deswegen weiterhin der UDP-Echo-Port angesprochen. Mit den **echo**- und **icmp (no data)**-Antworten zusammen erreicht man immerhin ca. 65 % der Hosts.

5.5.6 Probleme der ICMP-Generierung unter Solaris

Das Betriebssystem Solaris, das gerade für Internet-Server nicht selten eingesetzt wird, zeichnet sich durch einige Besonderheiten in seiner Implementierung von TCP/IP aus. Vom allem in Bezug auf die Generierung von ICMP-Paketen sind die Abweichungen vom Standard sind für die Messungen durchaus relevant. Auch andere Werkzeuge wie `traceroute` [38] oder `pathchar` [37] werden davon beeinflusst.

Zum einen erlaubt Solaris, eine als Kernelparameter einstellbare Zahl von zusätzlichen Datenbytes mit einer ICMP-Fehlermeldung zurückzuschicken. Eine nette Idee, die in der Praxis allerdings keine großen Vor- oder Nachteile mit sich bringt.

Zum anderen beschränkt Solaris die Zahl der generierten ICMP-Pakete auf zwei pro Sekunde. Treten in einer Sekunde mehr als zwei Ereignisse ein, die Anlass zur Erzeugung eines ICMP-Pakets bieten, so werden alle bis auf die ersten beiden ignoriert! Da dies auch noch unabhängig von der betroffenen Gegenstelle gilt, kann man sich bei einem Solaris-System nie darauf verlassen, dass jedes UDP-Paket an einen nicht bedienten Port mit einer ICMP-Meldung beantwortet wird. Man kann nicht einmal die Rate bestimmen, mit der solche Pakete ignoriert werden, solange man nicht sämtlichen Verkehr kennt, den das Solaris-System empfängt. Damit eignen sich solche Solaris-Maschinen definitiv nicht zur Bestimmung von Paketverlustraten, falls nicht der UDP-Echo-Port aktiviert ist.

Man kann davon ausgehen, dass die in Kapitel 5.5.5 als **icmp (data)** bezeichnete Klasse überwiegend aus Hosts mit dem Betriebssystem Solaris und deaktivierten UDP-Echo-Port besteht.

Kapitel 6

Messungen mit TCP-Syn-Paketen

6.1 Motivation

Ein nicht lösbares Problem der Messungen mit UDP-Echo-Paketen liegt darin, dass im Rahmen des gestiegenen Sicherheitsbedürfnisses heutzutage nicht nur viele Internet-Hosts den UDP-Echo-Dienst eingestellt haben, sondern dass entsprechende Pakete schon vor Erreichen des Hosts ausgefiltert werden. So kommt es, dass ein immer größer werdender Anteil aller Internet-Hosts für Messungen mit UDP-Echo-Paketen nicht mehr geeignet ist.

Das Ausfiltern der UDP-Echo-Pakete geschieht üblicherweise mit sogenannten „Screening Routern“ oder mit „Firewalls“. Ein Screening Router ist häufig ein handelsüblicher Router, auf dem allerdings bestimmte Features, sogenannte „Access-Control-Lists“ (ACL oder auch „Access-List“) aktiviert wurden. Diese betrachten jedes hereinkommende Paket und entscheiden anhand von relativ einfachen Regeln, die beispielsweise Adressen, Port- und Protokoll-Nummern auf bestimmte Werte prüfen, ob das Paket normal weitergeleitet werden soll oder nicht. Eine Firewall hingegen ist ein spezielles Gerät, meist auf Basis eines Multi-Homed Host, das nicht nur einzelne Pakete überprüft, sondern ganze Verbindungen nachvollzieht und dabei nur bestimmte Verbindungen zulässt. Dabei dient technisch gesehen die Firewall selber als Verbindungsendpunkt und leitet die Daten auf der Ebene einzelner Anwendungsprotokolle an die eigentlichen Server-Hosts weiter. Firewalls werden auch sehr häufig in Kombination mit Screening Routern eingesetzt, um die Sicherheitswirkung noch einmal zu erhöhen.

Beide Gerätetypen dienen letztlich dazu, aus dem Internet nur ganz bestimmte, erwünschte Daten zu einem Host durchzulassen. Da UDP-Echo-Pakete meist nicht zum „erwünschten“ Verkehr gehören, werden sie dann typischerweise ausgefiltert. Beim Ausfiltern wird häufig eine Fehlermeldung

in Form eines ICMP-Pakets vom Typ „communication prohibited“ erzeugt und an den Absendern geschickt. Leider sind diese Fehlermeldungen nicht zuverlässig, so dass sie in den Messungen nicht als vollwertiger Ersatz für UDP-Echo- oder ICMP-Port-Unreachable-Antworten zu gebrauchen sind.

Dieses Problem war Anlass, nach alternativen Meßmethoden zu suchen, die nicht auf bestimmte, „unübliche“ Dienste angewiesen sind und mit gängigen Firewalls und Access-Lists umgehen können.

Das im Internet heutzutage mit Abstand am häufigsten eingesetzte Transportprotokoll ist TCP. Die allermeisten der üblichen Internet-Dienste werden mit Hilfe von TCP erbracht. Die Kapitel 6.2 und 6.3 zeigen, wie der normale TCP-Verbindungsaufbau aussieht und wie man ihn sich für Messungen zunutze machen kann. Auf diese Art kann man einen beliebigen Host im Internet, der irgendeinen auf TCP basierenden Dienst anbietet, ansprechen und für Messungen verwenden. Solange man nur die TCP-Portnummer des angebotenen Dienstes verwendet, kann kein Screening Router und keine Firewall diese Messungen von einem normalen, erwünschten Verbindungsaufbau unterscheiden. Eine Firewall wird zwar den Verbindungsaufbau selber vornehmen, so dass man effektiv nur die Verbindung zur Firewall und nicht zum dahinterliegenden Host misst, aber da Firewalls üblicherweise wirklich in netztopologischer und räumlicher Nähe zu den zu schützenden Hosts aufgestellt werden, ist dies keine ernste Einschränkung.

Ein weiterer Vorteil der Messung mit TCP-Syn-Paketen ist in der sich abzeichnenden Tendenz zu **Quality of Service** (QoS) oder **Classes of Service** (CoS) im Internet zu sehen. In der Arbeitsgruppe DiffServ [34] der IETF werden zur Zeit Methoden entwickelt, mit denen künftig in Internet-Backbone verschiedene Dienstgüteklassen unterschieden werden können. Verschiedenen Klassen von IP-Paketen werden dabei in Backbone-Routern mit einem unterschiedlichen sogenannten „Per-Hop-Behaviour“ (PHB) behandelt, so dass je nach Klasse andere Ergebnisse für Paketverlustrate und Paketlaufzeit erzielt werden. Während das ganze Rahmenwerk noch in der Entwicklung ist, beherrschen bereits viele übliche Router einen Grundstock der notwendigen Features [81]. Wenngleich es noch nicht gängige Praxis ist, so beschäftigen sich doch viele Internet-Netzbetreiber schon mit dem Gedanken, mit Hilfe der DiffServ-Mechanismen UDP- und TCP-Verkehr in ihren Netzen getrennt voneinander zu behandeln. Messungen mit UDP-Paketen hätten dann nur eine eingeschränkte Aussagekraft für Dienste, die auf der Basis von TCP angeboten werden.

Kapitel 6.4 beschreibt die Implementierung eines Tools, das Messungen nach dem vorgeschlagenen Verfahren durchführen kann. Kapitel 6.5 beschäftigt sich mit einem speziellen Problem bei der Auswertung dieser Messungen, stellt eine Lösungsmöglichkeit vor und analysiert diese. Kapitel 6.6 schließlich analysiert die Brauchbarkeit der Messmethode insgesamt.

6.2 Aufbau einer TCP-Verbindung

TCP verwendet beim Verbindungsaufbau ein **Client/Server**-Prinzip. Der Server ist dabei ein Host, der typischerweise einen oder mehrere Dienste anbietet. Er wartet „passiv“ auf Anforderungen zum Verbindungsaufbau. Der Client hingegen ist der Host, der „aktiv“ zu einem vom ihm gewählten Zeitpunkt eine Verbindung initiiert. Der Aufbau einer TCP-Verbindung geschieht dabei über den sogenannten **three-way handshake** [77]. Der Name rührt daher, dass im Normalfall der Austausch von drei Paketen nötig ist, um eine TCP-Verbindung vollständig aufzubauen. Danach können in beiden Richtungen unabhängig voneinander Daten ausgetauscht werden. Der Abbau der Verbindung kann von einer beliebigen Seite eingeleitet werden und benötigt insgesamt weitere 3 bis 4 Pakete.

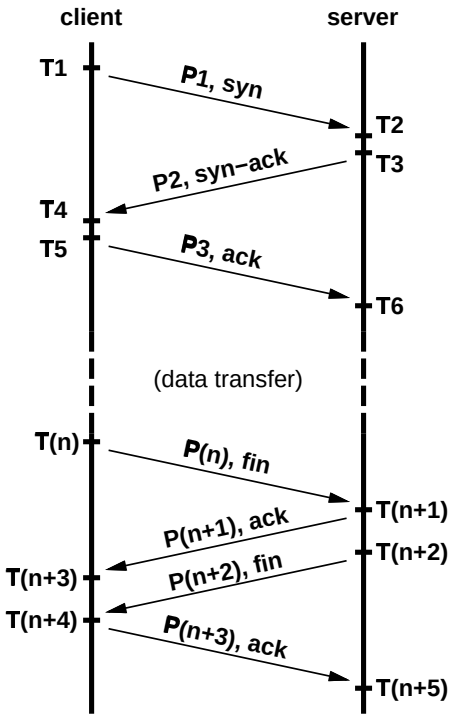

Abbildung 6.1: Ordnungsgemäßer Auf- und Abbau einer TCP-Verbindung.

Abbildung 6.1 zeigt den typischen Verlauf des Auf- und Abbaus einer TCP-Verbindung. Auf von oben nach unten verlaufenden Zeitachsen sind links die Ereignisse des Client-Hosts und rechts die des Server-Hosts dargestellt. Dazwischen sind die übertragenen Pakete angedeutet. Die Zeitpunkte der Ereignisse werden mit $T_1, T_2, ...T_n$ bezeichnet. Die Pakete werden durch Angabe der wichtigsten im TCP-Header gesetzten Flags gekennzeichnet, also etwa Syn, Ack, Fin.

Da für die vorzustellenden Messverfahren der Verbindungs-*Aufbau* von besonderem Interesse ist, wird er hier ausführlicher dargestellt:

T_1 Der Client initiiert die Verbindung, indem er ein Syn-Paket P_1 (ein TCP-Paket mit gesetztem „syn“-Flag) zum Server schickt.

T_2 Das Syn-Paket kommt beim Server an.

T_3 Unter der Voraussetzung, dass der angesprochene Dienst verfügbar ist, das heißt, dass ein Dienstprogramm auf eingehende Verbindungen wartet, wird mit einem Syn-Ack-Paket P_2 geantwortet. Dadurch wird das Syn-Paket des Client bestätigt und die Öffnung der Gegenrichtung der Verbindung wird eingeleitet.

T_4 Das Syn-Ack-Paket P_2 wird vom Client empfangen.

T_5 Der Client antwortet umgehend mit einem Ack-PaketP_3, das das Syn-Ack-Paket P_2 bestätigt. Aus Sicht des Client ist die Verbindung damit vollständig geöffnet und es können beliebige Daten in weiteren Paketen gesendet werden.

T_6 Der Server erhält das Ack-Paket P_3. Damit ist auch aus Sicht des Server die Verbindung geöffnet werden und auch der Server kann nun Daten senden.

$T_n..T_{n+5}$ In diesem Beispiel leitet der Client den Abbau der Verbindung mit einem Fin-Paket ein, der Server bestätigt und baut seinerseits die andere Richtung der Verbindung ab, was schließlich vom Client noch einmal bestätigt wird. Je nach Implementierung können P_{n+1} und P_{n+2} zu einem Paket zusammengefasst werden; der hier dargestellte Ablauf ist aber auf jeden Fall üblich und typisch.

Die Vorgänge von T_1 bis T_6 finden typischerweise im Kernel des Betriebssystems des jeweiligen Hosts statt. Erst bei T_5 beziehungsweise bei T_6 wird eine Systemfunktion beendet. Für diese Vorgänge innerhalb des Kernels brauchen Hosts in aller Regel sehr wenig Zeit im Vergleich zu Paketlaufzeiten im Weitverkehrsbereich, so dass die Bearbeitungszeit im Server $T_{P,Server} = T_3 - T_2$ sowie die Bearbeitungszeit im Client $T_{P,Client} = T_5 - T_4$ vernachlässigbar sind. Daher werden im folgenden T_2 und T_3 sowie T_4 und T_5 jeweils zu einem Zeitpunkt zusammengefasst.

6.3 Messungen

Bereits der Austausch der Pakete P_1 und P_2 reicht aus, um (bidirektionale) Paketlaufzeit und Paketverlustrate zu messen. Unter Vernachlässigung von $T_{P,Server}$ gilt

$$T_4 = T_1 + RTT + T_{P,Server} \tag{6.1}$$

$$\overset{T_{P,Server} \ll RTT}{\Rightarrow} RTT \approx T_4 - T_1 \tag{6.2}$$

Da sowohl T_4 als auch T_1 auf der Seite des Client messbar sind, kann damit die Paketlaufzeit bestimmt werden. Die bidirektionale Paketverlustrate lässt sich durch die Beobachtung bestimmen, wie häufig nach dem Aussenden eines geeigneten P_1 ein P_2 empfangen wird.

Zur Zeit T_4 liegen also alle für die Messung erforderlichen Informationen vor. Allerdings besteht zu der Zeit eine halb geöffnete Verbindung auf der Seite des Server. Würde man eine Reihe solcher Messungen zu einem bestimmten Host durchführen, so würden sich an diesem Host halboffene Verbindungen ansammeln, immer mehr Ressourcen belegen und möglicherweise den Server blockieren. Dies entspräche einem sogenannten **Denial-of-Service-Attack**, also einem Angriff, der auf das Behindern oder Ausschalten eines Dienstes abzielt. Ein auf vielen isolierten TCP-Syn-Paketen beruhender Denial-of-Service-Attack wird speziell als **TCP-Syn-Flooding** bezeichnet. Um zu verhindern, unbeabsichtigt einen solchen Angriff durchzuführen, muss darauf geachtet werden, dass jegliche Ressourcen auf Seiten des Server umgehend wieder freigegeben werden können. Dazu muss die halb aufgebaute Verbindung wieder abgebaut werden. Dies könnte geschehen, indem die TCP-Verbindung vollständig auf- und dann ordnungsgemäß abgebaut wird. Wie aus Abbildung 6.1 deutlich wird, würde das allerdings vergleichsweise viel Netzverkehr erzeugen. Statt dessen reicht es auch, ein geeignetes TCP-Paket mit gesetzten „RST“-Flag (für „Reset“) zu senden. Ein solches TCP-RST-Paket dient generell dazu, eine TCP-Verbindung aus jeglichem Zustand herauszuführen und abzubauen. Unter der Voraussetzung, dass der Server mit einem dort angebotenen Dienst angesprochen wird, hat eine Messung daher den in Abbildung 6.2 dargestellten Verlauf.

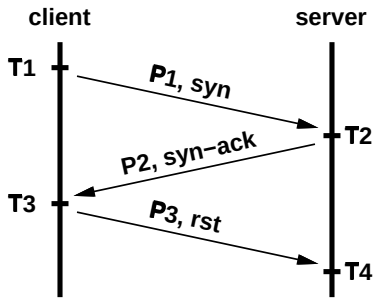

Abbildung 6.2: Messung durch unvollständigen Aufbau einer TCP-Verbindung.

Falls der Server mit einem *nicht* angebotenen Dienst angesprochen wird, so sendet er seinerseits umgehend ein TCP-Rst-Paket, mit dem die Nicht-Verfügbarkeit des angesprochenen Dienstes signalisiert wird [77]. Auch dieser Vorgang findet im Kernel statt, so dass die Bearbeitungszeit vernachlässigt werden kann. Das TCP-Rst-Paket kann für Messungen genauso gut verwendet werden, wie ein Syn-Ack-Paket. Es hat sogar den Vorteil, dass kein drittes Paket gesendet werden muss, da vom Server gar kein Verbindungsaufbau begonnen wurde, der abgebrochen werden müsste. Die Messung sieht dann aus wie in Abbildung 6.3 dargestellt.

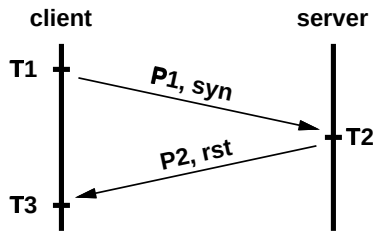

Abbildung 6.3: Messung durch Ansprechen einen nicht angebotenen TCP-Dienstes.

In beiden Fällen (der Dienst wird angeboten oder nicht, Abb. 6.2 und 6.3) gilt

$$RTT = T_3 - T_1 \tag{6.3}$$

(unter Berücksichtigung der in Kapitel 6.2 gemachten Vereinfachung).

6.4 Implementierung

Für die Implementierung der TCP-Syn-basierten Messmethode wurde eine ähnliche Struktur gewählt, wie bei der verbesserten Methode der UDP-Messungen (siehe Kapitel 5.4). Sie zerfällt in zwei Komponenten: eine aktive, die die gewünschten Messpakete erzeugt und versendet, und eine passive, die die versendeten und ankommenden Pakete beobachtet, einander zuordnet und die relevanten Zeitpunkte notiert.

Analog zu den UDP-Messungen ist es auch hier möglich, beim Einsatz eines geteilten Netzmediums aktive und passive Komponente auf verschiedene Hosts zu verteilen. Oder man kann beide Komponenten in einem Host vereinigen, was dann auch den Einsatz eines exklusiven Mediums erlaubt. Abbildung 6.4 zeigt die Struktur der TCP-Syn-basierten Messungen beim Einsatz eines exklusiven Mediums.

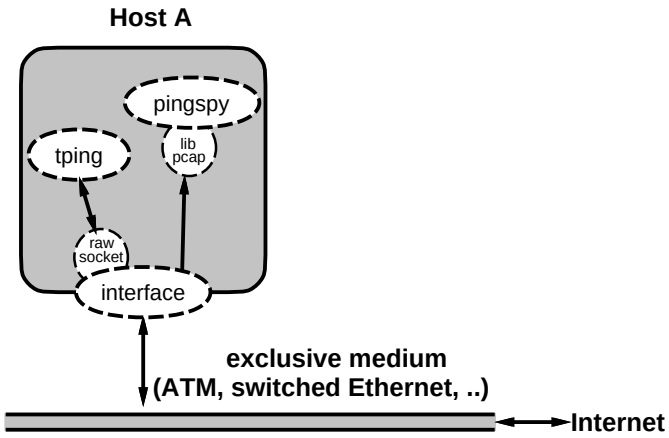

Abbildung 6.4: Struktur von `tping` und `pingspy` beim Einsatz eines exklusiven Mediums.

6.4.1 Sender

Die aktive Komponente, der Sender, wurde `tping` genannt. Sie erhält die für die Messungen benötigten Parameter (IP-Adressen A_1 und A_2, Senderate λ), berechnet (pseudo-)zufällige Startzeiten und erzeugt geeignete TCP-Syn-Pakete. Dabei wird ein Feld des TCP-Header, die Sequenz-Nummer, so generiert, dass der passiven Komponente eine eindeutige Zuordnung der möglichen Antworten erleichtert wird.

Um derartige Pakete erzeugen zu können, muss unter Unix ein sogenannter Raw-Socket verwendet werden [79]. Unter Java ist die Verwendung von Raw-Sockets ohne weiteres nicht möglich. Daher wurde für die Implementierung des Programms `tping` die Programmiersprache ANSI-C gewählt.

Die Kommandozeile, über die die benötigten Parameter übergeben werden, sieht wie folgt aus:

```
usage: tping [ options ] { hosts }
options: -f file   read hosts from file
         -p port   specify port to send to
         -n num    number of probes to each host
         -r lambda set rate (for each host)
         -t num    number of hosts to process in parallel
         -T ttl    IP TTL to use
         -l ip     local ip address
```

Die IP-Adressen der Hosts, zu denen Messpakete geschickt werden sollen, können sowohl direkt in der Kommandozeile angegeben werden, als auch aus einer Datei gelesen werden. In beiden Fällen kann der TCP-Port des Ziel-Hosts angegeben werden. Als weitere Bestandteile der zu erzeugen TCP-

Syn-Pakete lassen sich die TTL angeben und die IP-Adresse des absendenden Hosts („local ip address“). Außerdem kann angegeben werden, wie viele Pakete zu jedem Host gesendet werden sollen und mit welcher Rate. Falls nichts anderes angegeben wird, werden pro Host 10 Pakete mit einer Rate von einem Paket pro Sekunde verschickt. Als IP-Adresse des absendenden Hosts wird dann die erste gefundene tatsächliche Adresse des lokalen Systems verwendet.

Die Angabe der „local ip address“ erlaubt es in dem Fall, dass ein Multi-Homed-Host für die Messungen verwendet wird, eine bestimmte der eigenen IP-Adressen für die Messungen zu verwenden. Dies erleichtert es, eine IP-Adressen speziell den Messungen zu widmen und die in Kapitel 4.4 erwähnten Maßnahmen durchzuführen.

Technisch wäre es auch möglich, hier eine ganz andere IP-Adresse anzugeben, etwa eine, unter der gar kein Host existiert. In diesem Fall würde allerdings genau der in Kapitel 6.3 beschriebene Denial-Of-Service-Angriff TCP-Syn-Flooding stattfinden. Nur dadurch, dass die Syn-Ack-Pakete normalerweise im TCP-Protokoll des Mess-Rechners ankommen, dort aber nicht mit einer ordentlichen TCP-Verbindung in Bezug gebracht werden können, wird ein TCP-RST-Paket erzeugt, das schließlich im Ziel-Host die halb aufgebaute TCP-Verbindung wieder abbaut.

Das Programm `tping` dient nur zum Versenden von Paketen und unternimmt seinerseits keinerlei Versuch, Pakete zu empfangen oder auszuwerten. Daher erzeugt es im Normalfall keine Ausgabe.

6.4.2 Empfänger

Die passive Komponente, der Empfänger, behielt ihren Namen `pingspy`, wurde allerdings komplett neu implementiert. Von der früheren Methode, das Programm `tcpdump` als Schnittstelle zum Datennetz zu verwenden, wurde Abstand genommen. Zum einen müssen zur Zuordnung der TCP-Pakete viele Daten-Felder der Pakete angesehen werden, bei denen `tcpdump` keine große Hilfe ist, und zum anderen erwies sich die Implementierung von `pingspy` in der Programmiersprache Perl schon bei den UDP-Messungen als nennenswerter Verbraucher von CPU-Zeit, was vor allem auf das umständliche Auswerten der Ausgabe von `tcpdump` zurückzuführen ist.

Die Neuimplementierung wurde objektorientiert in der Programmiersprache C++ vorgenommen, wobei Datenstrukturen der Standard Template Library [55] zum Einsatz kamen. Als Interface zum Netz diente die Bibliothek `libpcap` [50], die direkt auf Betriebssystem-spezifische Schnittstellen zum Netz-Interface zugreift, ihrerseits aber eine Betriebssystemunabhängige Schnittstelle zum Lesen beliebiger Pakete anbietet.

Die Kommandozeile sieht wie folgt aus:

```
usage: ./pingspy [d] [c<cmd>] [i<interface>] [l<local ip>]
                 [n<num probes>] [q<quit time>]
```

Mit „d“ kann ein Debug-Modus aktiviert werden. Durch die Angabe von „cmd“ und „interface“ können eingebaut Voreinstellungen zur Ansteuerung der `libpcap` geändert werden: mit „cmd“ kann in der `libpcap`-eigenen Syntax genauer spezifiziert werden, welche Pakete schon in unteren Schichten gefiltert werden, und mit „interface“ kann angegeben werden, welches Netz-Interface zur Beobachtung der Verkehrs verwendet werden soll, falls mehrere Netz-Interfaces zu Verfügung stehen. Mit „local ip“ kann festgelegt werden, welche IP-Adressen als lokal angesehen werden. Dabei kann neben einer IP-Adresse auch eine Netzmaske angegeben werden, so dass ein ganzer Bereich von IP-Adressen verwendet wird. Alle anderen IP-Adressen werden als potenzielle Ziel-Hosts gewertet; jedes TCP-Syn-Paket mit einem lokalen Absender und einem nicht-lokalen Ziel wird als potenzielles Messpaket gewertet. Die Angabe von „num probes“ sagt dem Programm, wie viele Messpakete pro Ziel-Host mindestens beobachtet werden müssen, damit die Ausgabe einer Messreihe sinnvoll ist. Mit „quit time“ schließlich kann die Laufzeit des Programm begrenzt werden. Dies ist deswegen sinnvoll, weil bei der typischen Verwendung das Programm `pingspy` im Hintergrund gestartet wird, um gleichzeitig im Vordergrund die aktiven Messungen mit dem Programm `tping` zu initiieren. Die Angabe der „quit time“ stellt dann das Beenden des Hintergrund-Prozesses zur richtigen Zeit sicher.

Bei Beendigung erzeugt `pingspy` schließlich eine Ausgabe der folgenden Art (stark gekürztes Beispiel):

```
# icmp type 3, code 1 from 205.160.25.4 to 134.95.129.25 concerning 205.160.26.44
# icmp type 3, code 1 from 205.160.25.4 to 134.95.129.25 concerning 205.160.26.44
# icmp type 3, code 1 from 205.160.25.4 to 134.95.129.25 concerning 205.160.26.44
# icmp type 11, code 0 from 145.253.1.184 to 134.95.129.25 concerning 145.253.94.123
206.132.234.101: 0 109.6 A 27701.4 117.1 A 130126 111 A 137097 114.3 A 297734 111.2 A
206.132.234.218: 0 112.1 A 24661.2 0 L 49392.6 109.5 A 67913.3 110.6 A 111545 0 L
206.165.251.235: 0 159.3 R 22851.3 159.1 R 61003.1 162.2 R 74923.8 158.9 R 107895 160.3 R
205.160.26.44: 0 110 ? 9850.6 110.5 ? 80164.1 112.3 ? 86564.3 109.7 ? 94754.7 108.8 ?
145.253.94.123: 0 161.7 R 14550.6 233 R 89304.1 159.8 R 99984.6 210.1 ? 131026 130.2 R
```

Zunächst werden alle empfangenen ICMP-Pakete dokumentiert. Dann erfolgt für jeden Ziel-Host die Ausgabe der Messreihe mit Angabe der IP-Adresse des Host und einem Tripel für jede einzelne Messung:

- Die Startzeit relativ zum Beginn der Messreihe.
- Die bidirektionale Paketlaufzeit, oder 0 im Falle eines Paketverlusts.
- Die Art der Antwort: `A` für ein TCP-Syn-Ack-Paket, `R` für ein TCP-RST-Paket, `?` für ein ICMP-Paket und `L` für einen Paketverlust (keine Antwort während der Programmlaufzeit). Falls mehr als eine Antwort pro Messpaket eingeht, wird allen anderen Antworten als der ersten ein `d` (für Doppelübertragung) vorangestellt.

Alle Zeitangaben verstehen sich in Millisekunden mit einer Auflösung von Zehntel-Millisekunden.

6.4.3 Rein passive Messungen

Die Implementierung von `pingspy` erlaubt auch rein passive Messungen ohne den Einsatz von `tping`: „local ip“ muss so gewählt werden, dass solche Hosts als lokal betrachtet werden, die häufig genug TCP-Verbindungen aufbauen, und es muss sichergestellt werden, dass dieser Verkehr von `pingspy` beobachtet werden kann, etwa durch den Einsatz eines geteilten Netzmediums oder indem `pingspy` direkt auf einem Host läuft, der viele Verbindungen aufbaut. Für jede ohnehin aufgebaute TCP-Verbindung ergibt sich für `pingspy` dann *eine* Messung. Aus den im Rahmen einer bestehenden TCP-Verbindung übertragenen Daten kann `pingspy` allerdings keine weiteren Messwerte gewinnen.

Auch mit diesem Ansatz wurden Tests durchgeführt. Zum Einsatz kam dabei der zentrale WWW-Proxy-Server der Universität zu Köln, der naturgemäß viel Benutzer-Verkehr bündelt und der viele TCP-Verbindungen aufbaut, die meist nur von kurzer Dauer sind.

Bei rein passiven Messungen ist man auf das Verhalten der Benutzer angewiesen und dies hat, wie sich schnell zeigte, eine ungünstige Verteilung: zu bestimmten Tageszeiten werden relativ wenige Server sehr häufig angesprochen und andere nur selten. Zu anderen Tageszeiten werden die gleichen Server dann wiederum kaum noch angesprochen, so dass sich schlecht aussagekräftige Vergleiche mit einer kontrollierten List von Internet-Hosts durchführen lassen. Darüber hinaus bedeutet der Einsatz eines Paketfilters immer auch eine potenzielle Einschränkung der Leistungsfähigkeit eines Computer-Systems und da der betroffene WWW-Proxy-Server mittlerweile kaum mehr über ungenutzte Leistungsreserven verfügt, wurde von dieser Methode nach kurzen Tests wieder Abstand genommen.

6.5 Eliminierung von TCP-Neuübertragungen

6.5.1 Entstehung der TCP-Neuübertragung

Als zuverlässiges Übertragungsprotokoll beinhaltet TCP Methoden, Paketverluste auszugleichen. Dafür werden unter anderem sogenannte **Retransmission Timer** verwendet, die für die Neuübertragung eines Pakets sorgen, falls innerhalb eines bestimmtes Zeitintervalls keine Bestätigung für ein gesendetes Paket empfangen wurde. In diesem Fall wird exakt das gleiche Paket nach Ablauf des zuständigen Retransmission Timers neu gesendet. Diesen Vorgang wird als Neuübertragung oder **Retransmission** bezeichnet.

Dieser Mechanismus kommt bereits beim Aufbau einer TCP-Verbindung zum Einsatz (siehe Kapitel 6.2). Paket P_1 wird vom Messprogramm erzeugt und unterliegt daher nicht dem Neuübertragungsverfahren von TCP. Wenn P_1 auf dem Weg zum Ziel verloren geht, kann dort keinerlei Aktion veranlasst werden und da auch das Messprogramm keine Neuübertragung durchführt,

kann durch das Ausbleiben jeglicher Reaktion zuverlässig auf einen Paketverlust geschlossen werden.

Paket P_2 jedoch wird von der TCP-Implementierung des angesprochenen Host generiert und bei Bedarf neu übertragen. Abbildung 6.5 zeigt den Ablauf für den Fall, dass der erste Versuch, P_2 zu übertragen, fehlschlägt:

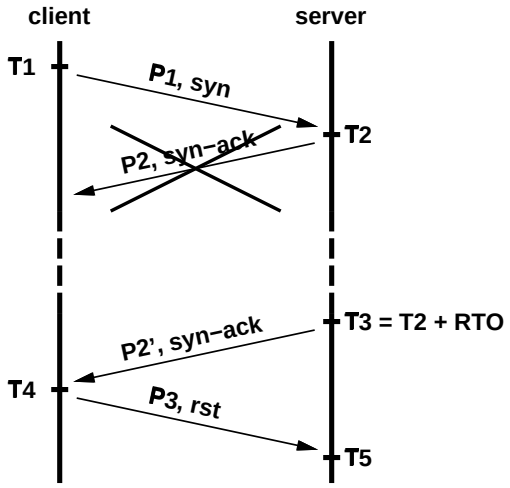

Abbildung 6.5: Verlust und Neuübertragung des 2. Pakets.

T_1 Der Client sendet das verbindungseröffnende Syn-Paket.

T_2 Der Server bestätigt den Verbindungsaufbau mit einem Syn-Ack-Paket. Dieses Paket geht auf dem Weg zum Client verloren.

T_3 Der Server wartet für ein bestimmtes Zeitintervall, genannt RTO („Retransmission Timeout“), auf eine Bestätigung des Verbindungsaufbaus oder auf einen Abbruch der Verbindung. Zur Zeit $T_3 = T_2 + RTO$ hat er weder das eine noch das andere empfangen und sendet Paket P_2 erneut. Dies wird als P_2' bezeichnet.

T_4 Nun kommt Paket P_2' beim Client an. Es wird ein Rst-Paket verschickt, um die halb aufgebaute Verbindung beim Server wieder abzubauen.

T_5 Der Server empfängt das rst-Paket und baut die Verbindung ab. Dementsprechend werden im folgenden auch keine weiteren Neuübertragungen von P_2 mehr unternommen.

Das Problem liegt nun darin, dass P_2 und P_2' aus genau der gleichen Bitfolge bestehen und daher für den Client nicht unterscheidbar sind. Zur Zeit T_4 kann er daher nicht erkennen, dass es sich bei P_2' um eine Neuübertragung

handelt. Eigentlich sollte ein Paketverlust gemeldet werden, da ja P_2 verloren gegangen war. Statt dessen wird eine erfolgreiche Übertragung gemeldet und als Paketlaufzeit wird fälschlicherweise $RTT' = T_4 - T_1 = RTT + RTO$ angenommen! RTO ist nun keinesfalls klein gegenüber RTT, sondern normalerweise deutlich größer (s.u.), so dass diese Neuübertragung sowohl die Messung der Paketverlustrate, als auch die der Paketlaufzeit deutlich verfälscht.

6.5.2 Berechnung des Retransmission Timeout

Das TCP-RFC [67] besagt, dass RTO dynamisch bestimmt wird und schlägt ein Formel vor, nach der das geschehen kann. Dabei wird in einer TCP-Verbindung durch den Datenfluss permanent die Paketlaufzeit gemessen, geglättet und ein Vielfaches davon (das Doppelte) als Ausgangswert für RTO verwendet. Ziel ist es, RTO so zu bestimmen, dass es deutlich über der zu erwartenden Paketlaufzeit liegt, um nicht unnötige Neuübertragungen zu beginnen, die sich mit nennenswerter Wahrscheinlichkeit mit der Ankunft ausstehender Bestätigungen überschneiden. Erfahrungen im wachsenden Internet haben gezeigt, dass der ursprüngliche Vorschlag noch nicht ausreichend ist [39]. Statt dessen wurde eine neuen Formel empfohlen, die neben der geglätteten gemessenen Paketlaufzeit auch ein Vielfaches der ebenfalls geglätteten gemessenen Varianz der Paketlaufzeit beinhaltet:

$$RTO = A + 4\,D \tag{6.4}$$

wobei A die geglättete gemessene Paketlaufzeit und D die geglättete gemessene Varianz der Paketlaufzeit ist. Das „Host Requirements RFC“ [14] schreibt die Verwendung dieser Formel verbindlich für alle Internet-Hosts vor.

Ebenso schreibt es die Verwendung des sogenannten „exponential backoff“ für wiederholte Neuübertragungen vor. Dies wird so implementiert, dass das Zeitintervall zwischen zwei Neuübertragungen mit jeder Wiederholung verdoppelt wird. Wenn T_i der Zeitpunkt für den i-ten Versuch der Übertragung eines bestimmten Pakets ist, dann soll also für jedes $k \in N$ gelten

$$T_{k+1} - T_k = 2\,(T_k - T_{k-1}) \tag{6.5}$$

In der Praxis wird allerdings implementationsabhängig meist eine obere Grenze für das Zeitintervall zwischen zwei Neuübertragungen festgelegt, beispielsweise 64 Sekunden [77]. Ebenso gibt es eine implementationsabhängige Obergrenze für die Zahl der Versuche. Ist sie überschritten, so wird die TCP-Verbindung abgebrochen.

Dem Host Requirements RFC zufolge gelten für die Syn-Pakete des TCP-Verbindungsaufbaus die gleichen Regeln wie für die Neuübertragung von Datenpaketen. Allerdings kommt beim Verbindungsaufbau zum Tragen, dass

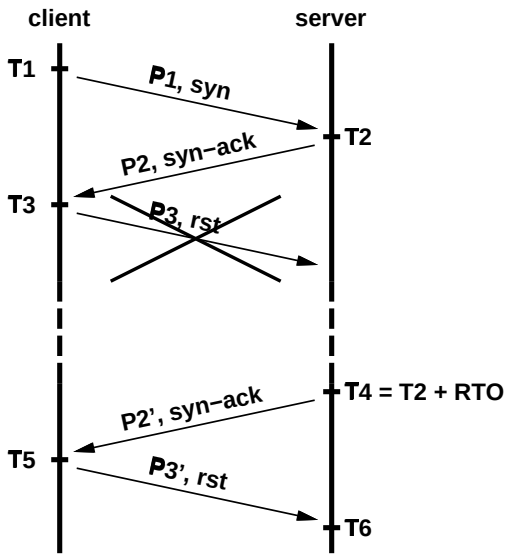

Abbildung 6.6: Verlust der Antwort des Client und Neuübertragung des 2. Pakets.

TCP noch keinen gemessenen Wert für die Paketlaufzeit A (und die Varianz D) vorliegen hat. Daher werden A und D zu Beginn einer Verbindung mit festen, implementationsabhängigen Werten belegt. Das RFC empfiehlt dafür den Wert $A_{initial} = 3\,\mathrm{s}$ und $D_{initial} = 0$.

6.5.3 Charakterisierung von Hosts

Zur Erforschung der Retransmission Timer kann man sich die Tatsache zu Nutze machen, dass Neuübertragungen nicht nur stattfinden, wenn das Syn-Ack-Paket des Server verloren geht, wie in Abbildung 6.5, sondern auch dann, wenn das darauf folgende Ack- oder Rst-Paket des Client verloren geht. Abbildung 6.6 zeigt diesen Fall. Der Client hat hier mit Paket P_2 eine gültige Antwort bekommen. Da der Server aber das rst-Paket nicht erhält, überträgt er das Paket neu. Der Client kann P_2' nun als Neuübertragung erkennen und relativ genau den Wert RTO des Server bestimmen: $RTO = T_4 - T_2 \approx T_5 - T_3$.

Um zu sehen, wie sich Hosts im Internet heutzutage tatsächlich verhalten, wurde die Verteilung der Paketlaufzeit bei umfangreichen Messungen untersucht. Abbildung 6.7 zeigt die Verteilung der Paketlaufzeit bei einer Messung mit UDP-Echo-Paketen im Vergleich zu einer Messung mit TCP-Syn-Paketen. Bei den TCP-Syn-Paketen wurden die erkennbaren Neuübertragungen und die nicht als Neuübertragung erkannten Antworten unterschieden und getrennt aufgezeichnet. Grundlage dieser Abbildung sind

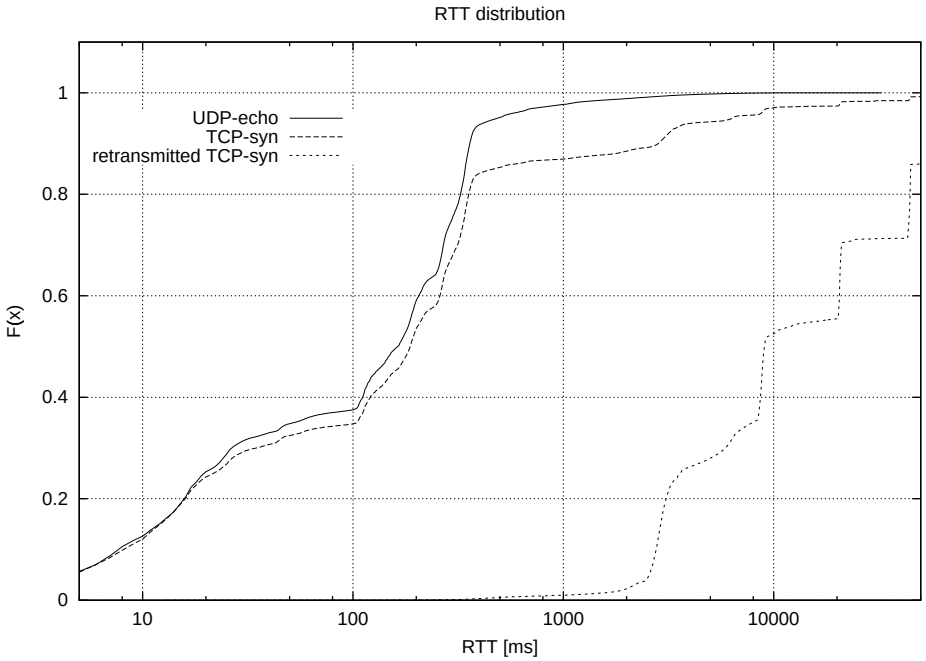

Abbildung 6.7: Verteilung der Paketlaufzeit bei Messung mit UDP-Echo- und TCP-Syn-Paketen.

Messungen, die in der 28. Kalenderwoche 1999 zu 250 weltweit verteilten Internet-Hosts durchgeführt wurden. Dabei wurde 7 Tage lang zu jedem Host stündlich eine Messung mit 20 Testpaketen durchgeführt.

Die Untersuchung zeigt, dass sich der größte Teil aller Hosts an die Empfehlung $A_{initial} = 3\,\mathrm{s}$ des Host Requirements RFC hält, dass einige jedoch andere Werte verwenden. Abbildung 6.8 gibt einen noch genaueren Blick auf die Verteilung der Art der Hosts. Sie zeigt die Verteilung der Hosts nach der schnellsten für jeden Host gemessenen Neuübertragung. Ein nennenswerter Anteil sendet Neuübertragungen mit einer Verzögerung von ca. 2.5 bis 3 Sekunden. Das deutet durchaus auch auf einen Wert von $A_{initial} = 3\,\mathrm{s}$ hin, wobei allerdings ein grober Timer mit einer Auflösung von 500 Millisekunden verwendet und nach unten abrundet wird, wie er für eine bestimmte TCP-Implementierung üblich ist (der „slow timer“ von BSD-Unix [88]). Einige Hosts verwenden schließlich Werte, die wie zufällig aussehen und teilweise sehr klein sind. Hier liegt die Vermutung nahe, dass diese Hosts von der Möglichkeit Gebrauch machen, die Messwerte für A und D außerhalb einzelner TCP-Verbindungen systemweit zusammen mit Routing-Tabellen abzulegen, um bei neuaufzubauenden Verbindungen auf die Messwerte früherer Verbindungen zurückzugreifen [77].

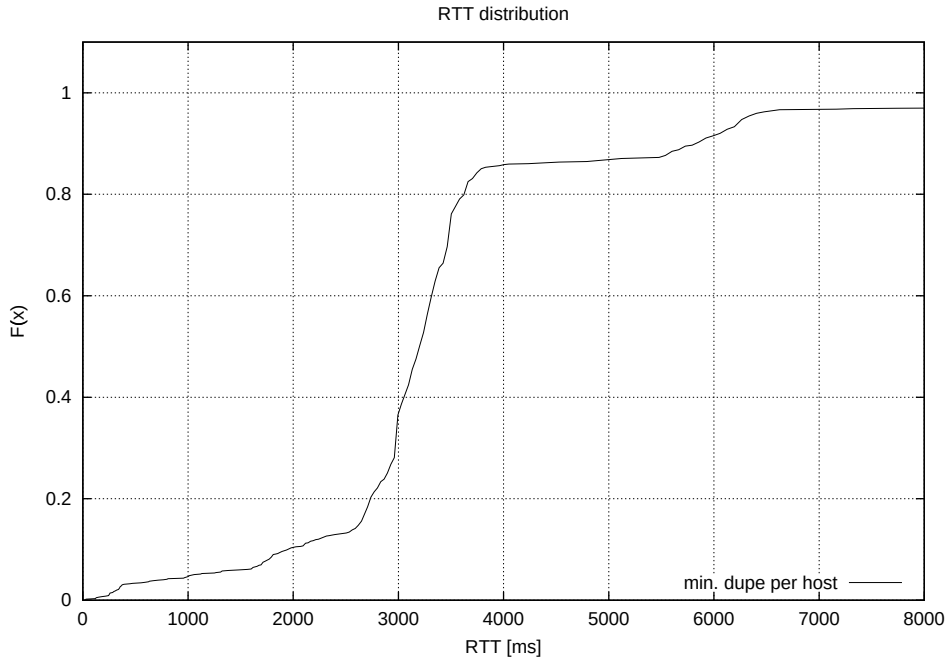

Abbildung 6.8: Verteilung der frühesten beobachteten Neuübertragungen.

6.5.4 Korrektur der RTT-Verteilung

Das in Kapitel 6.5.1 angesprochene Problem, dass der Client nicht zwischen Erst- und Neuübertragung des TCP-Syn-Ack-Pakets unterscheiden kann, lässt sich nun in den Griff bekommen, wenn man annimmt, dass jeder Host einen charakteristischen Wert für $A_{initial}$ implementiert, diesen Wert für alle zu untersuchenden Hosts bestimmt und alle gemessenen Paketlaufzeiten, die $A_{initial}$ überschreiten, verwirft.

Das funktioniert solange zufriedenstellend, wie $A_{initial}$ tatsächlich deutlich größer als RTT ist. In Abbildung 6.7 ist neben den Ergebnissen der TCP-Syn-Messungen auch eine Kurve für die Ergebnisse der UDP-Echo-Messungen eingetragen. Sie dient hier als Maßstab für die nicht durch TCP-Neuübertragungen verfälschte Verteilung von Paketlaufzeiten im Internet: mehr als 99 % aller beobachteten Paketlaufzeiten liegt unter zwei Sekunden und 98 % liegen unter einer Sekunde. Was so schon einigermaßen brauchbar aussieht, verbessert sich weiter, wenn man nicht die Summe vieler Hosts sondern einzelne Hosts für sich genommen betrachtet. Viele Hosts, die einen kleinen Wert für $A_{initial}$ verwenden, haben auch sehr kurze Paketlaufzeiten, so dass die RTT-Verteilung hier häufig schon ohne Verluste bei einer Sekunden abgeschnitten werden kann. Nur bei wenigen Hosts geht die Häufung der offensichtlichen Erstübertragung in die der Neuübertragungen über.

Diese Hosts müssen allerdings erkannt und von den Messungen ausgeschlossen werden, um keine nennenswerte Verfälschung des Gesamtergebnisses zu riskieren!

6.5.5 Heuristische Ermittlung des Retransmission-Timeout

Ein weiteres Problem taucht auf, wenn man $A_{initial}$ für die Gesamtheit aller zu untersuchenden Hosts ermitteln will: ein Teil der Hosts offenbart sein Verhalten in Bezug auf $A_{initial}$ durch erkennbare Neuübertragungen, der andere Teil tut dies nicht. Dass der letztgenannte Teil leider nicht vernachlässigbar klein ist, liegt daran, dass die Pfade im Internet häufig asymmetrisch sind [62] und dass sich das auch auf die Paketverluste auswirkt. Wenn die unidirektionale Paketverlustrate auf dem Hinweg zu einem Host sehr niedrig, auf dem Rückweg aber sehr hoch ist, dann treten die Paketverluste praktisch nur nach dem Muster von Abbildung 6.5 auf und nicht nach dem Muster von Abbildung 6.6. Es fehlen dann die erkennbaren Neuübertragungen, die Rückschlüsse auf $A_{initial}$ zulassen.

Um dieses Problem zu lösen, wurde ein heuristischer Algorithmus entwickelt, der die RTT-Verteilung eines Hosts analysiert und eine Abschätzung für $A_{initial}$ so bestimmt, dass die Häufung der echten Erstübertragungen von der Häufung der Neuübertragungen getrennt wird.

Ausgangspunkt für die Heuristik ist ein einfaches analytisches Modell der RTT-Verteilung. Die Parameter dieses Modells werden an die beobachtete RTT-Verteilung angepasst und schließlich wird das so parametrisierte Modell mit einem Histogramm der gemessenen Werte verglichen, um auf diesem Vergleich aufbauend die Neuübertragungen als deutliche Abweichung gut erkennen zu können.

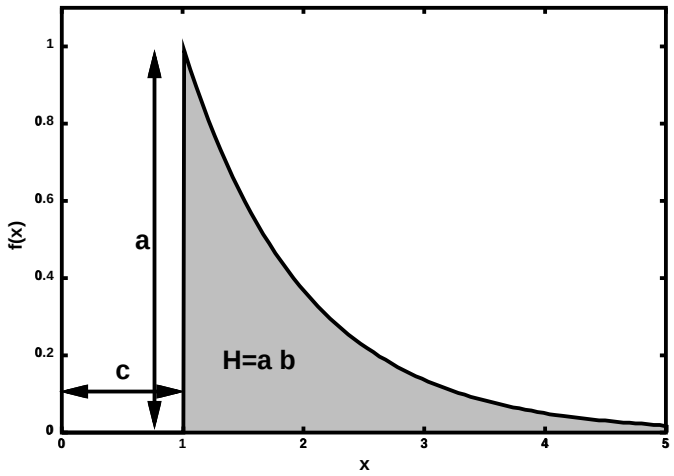

Abbildung 6.9: Negativ-exponentielle Verteilung.

Als Modell wurde die negativ-exponentielle Verteilung gewählt. Sie ist in der Warteschlangentheorie wegen vieler gutmütiger Eigenschaften sehr beliebt. Die Realität der meisten Internet-Größen beschreibt sie nur unvollständig, ist hier als grobe Näherung aber absolut ausreichend. Sie ist durch die folgende Funktion gegeben:

$$f(x) = \begin{cases} 0 & \text{für} \quad x < c \\ a\, e^{\frac{c-x}{b}} & \text{für} \quad x \geq c \end{cases} \tag{6.6}$$

Dabei steht x für die Paketlaufzeit, $f(x)$ beschreibt die Wahrscheinlichkeitsdichte in Abhängigkeit von $RTT = x$ und a, b und c sind die Parameter des Modells. Abbildung 6.9 zeigt den Graph der Funktion. Ihre Parameter kann man wie folgt deuten:

a ist das Maximum der Wahrscheinlichkeitsdichte.

b beeinflusst die Fläche H unter der Kurve:

$$H = \int_{x=0}^{\infty} f(x)\, dx = \int_{x=0}^{\infty} a\, e^{\frac{-x}{b}}\, dx = a\, b \tag{6.7}$$

Im diskretisierten Fall entspricht H der Zahl der Messungen.

c ist der Abstand der y-Achse vom Maximum. Netztechnisch lässt sich c als minimale Paketlaufzeit betrachten, die durch die Ausbreitungsgeschwindigkeit der Signale und notwendige Sende- und Bearbeitungszeiten gegeben ist. Jede Erhöhung dieser Paketlaufzeit kommt durch Wartezeiten in Routern zustande.

Der heuristische Algorithmus hat die Aufgabe, für jeden Host anhand der gemessenen Paketlaufzeiten eine maximale Paketlaufzeit RTT_{max} so zu bestimmen, dass RTT_{max} möglichst groß ist, aber eindeutig die Erstübertragungen von den Neuübertragungen trennt. Alle Messungen mit einer Paketlaufzeit $RTT \geq RTT_{max}$ werden dann bei der Auswertung als Paketverlust gewertet, alle Messungen mit einer Paketlaufzeit $RTT < RTT_{max}$ hingegen werden als gültig betrachtet. RTT_{max} soll nicht größer als 3 Sekunden werden, da nach dieser Zeit gemäß Host Requirements RFC [14] auf jeden Fall Neuübertragungen zu erwarten sind.

Der Algorithmus erzeugt zunächst ein Histogramm der gemessenen Paketlaufzeiten. Entsprechend der drei Parameter von $f(x)$ müssen dann genau drei unabhängige Größen gewonnen werden, um die Funktion zu parametrisieren. Die Verwendung der Zahl der Messwerte im Histogramm H und der kürzesten beobachteten Paketlaufzeit RTT_{min} ist naheliegend. Allerdings wird nicht etwa ein direkter Schätzwert für a als dritte Größe verwendet, sondern der Median RTT_{med}. Die Auflösung I_{hist} des Histogramms darf nämlich nicht zu fein gewählt werden, weil der Algorithmus sonst bei einer

kleineren bis mittleren Anzahl Messwerte zu fehleranfällig ist. Und durch die grobe Wahl für I_{hist} wird ein direkter Schätzwert für a stark verfälscht. Außerdem stimmt gerade an dieser Stelle das Modell häufig nicht besonders gut mit der Wirklichkeit überein. Der Median hingegen stellt sich als wesentlich robusteres Maß dar und liefert gute Ergebnisse. Der Algorithmus arbeitet nun wie folgt:

- Bestimme RTT_{min} und setze $c \leftarrow RTT_{min}$.
- Erzeuge ein Histogramm $H_0, H_1, \ldots H_{n-1}$ der gemessenen Paketlaufzeiten. Dabei ist H_i die Zahl aller Messungen, deren RTT im Intervall $[i \cdot I_{hist}, (i+1) \cdot I_{hist})$ liegt. I_{hist} wird zu $I_{hist} = 100\,\text{ms}$ und n, die Zahl der im Histogramm betrachteten Intervalle, zu $n = \frac{3\,\text{S}}{I_{hist}}$ gewählt.
- Bestimme die Zahl der Messwerte im Histogramm

$$H = \sum_{i=0}^{n-1} H_i \tag{6.8}$$

- Bestimme den Median RTT_{med} der Paketlaufzeiten. Um den Wert aus dem Histogramm zu ermitteln, wird eine ganze Zahl m so bestimmt, dass gilt

$$\sum_{i=0}^{m-1} H_i < \frac{1}{2} H \quad \wedge \quad \sum_{i=0}^{m} H_i \geq \frac{1}{2} H \tag{6.9}$$

  Dann kann der Median abgeschätzt werden durch

$$RTT_{med} \approx (m - 1/2)\, I_{hist} \tag{6.10}$$

- Wegen 6.7 und

$$\frac{1}{2} H = \frac{1}{2} a\, b = \int_c^{RTT_{med}} a\, e^{\frac{c-x}{b}}\, dx \tag{6.11}$$

$$\Leftrightarrow \qquad b = \frac{RTT_{med} - c}{\ln 2} \tag{6.12}$$

  setze $b \leftarrow \frac{RTT_{med} - c}{\ln 2}$.
- Setze $a \leftarrow \frac{H}{b}$.
- Schleife: beginne mit $i \leftarrow m$.
  - Vergleiche $f(x = i\, I_{hist})$ mit H_i.
    - Falls der Wert des Histogramms den des Modells deutlich unterschreitet ($f(x = i\, I_{hist}) < 0.1\, H_i$), vermerke diese Tatsache.

  - Falls der Wert des Histogramms den des Modells deutlich überschreitet ($f(x = i\,I_{hist}) > 1.4\,H_i$) und der Wert vorher schon einmal deutlich unterschritten wurde (s.o.), dann brich die Schleife ab und verwende $RTT_{max} = 0.9\,i\,I_{hist}$.
- Erhöhe i um eins und fahre mit der Schleife fort.

Dieser Algorithmus verwendet gewissermaßen eine Hysterese um das Modell, die einmal unter- und dann überschritten werden muss, damit eine Häufung von Messwerten als Neuübertragung erkannt wird. Für die Breite der Hysterese wurden empirisch sinnvolle Werte ermittelt. RTT_{max} wird zur Sicherheit um 10 % unter dem eigentlich ermittelten Wert angesetzt, weil bei Messungen von kleinerem oder mittlerem Umfang häufig nur wenige Neuübertragungen beobachtetet werden und sonst durch statistische Ungenauigkeiten leicht zu hohe Werte für RTT_{max} verwendet würden.

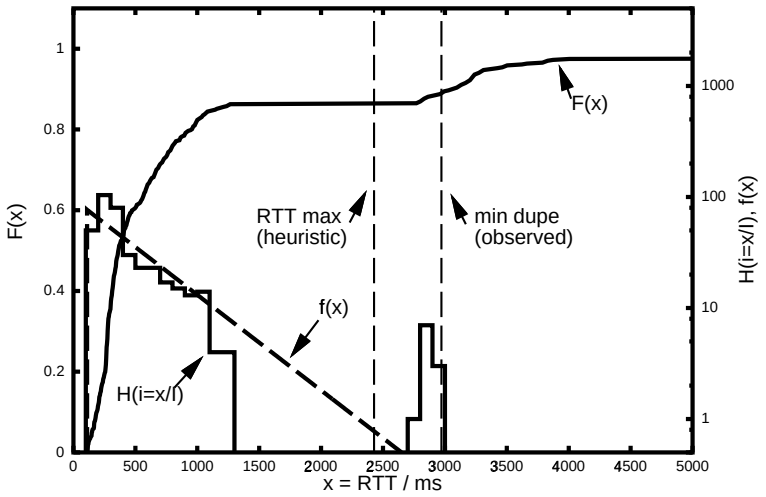

Abbildung 6.10: Beispiel für die Bestimmung der maximal zulässigen Paketlaufzeit mittels Heuristik.

Abbildung 6.10 veranschaulicht die Vorgehensweise des Algorithmus am Beispiel eines Hosts. Grundlage sind die Messungen eines Tages für diesen Host, in diesem Fall $24 \cdot 20 = 480$ Messwerte. Für die eingetragene Verteilungsfunktion $F(x)$ gilt die linke y-Skala, für das Histogramm H_i mit $i = \lfloor \frac{x}{I_{hist}} \rfloor$ und das Modell $f(x)$ die rechte y-Skala. Des weiteren ist durch vertikale Linien markiert, welchen Wert der Algorithmus als RTT_{max} gewählt hat und im Vergleich dazu die kürzeste als Neuübertragung erkannte Paketlaufzeit („min dupe“). Man sieht hier deutlich, dass die bloße Betrachtung der erkennbaren Neuübertragungen nicht ausreicht, um eine gute Trennung zwischen Erst- und Neuübertragungen zu ermöglichen. Der heuristische Algorithmus hingegen funktioniert erfahrungsgemäß gut. Der hier

als Beispiel ausgewählte Host zeichnet sich durch eine relativ große Varianz der Paketlaufzeit aus – die meisten anderen Hosts haben eine erheblich kleinere Varianz, was zu kompakteren Häufungen im Histogramm führt und vom vorgestellten Algorithmus noch leichter zu erkennen ist.

6.6 Analyse

6.6.1 Anteil der verwendbaren Hosts

Zur Beurteilung des heuristischen Algorithmus wurden die Messergebnisse für über 1000 weltweit verteilte Hosts ausgewertet. Die einzigen Fälle, in denen der Algorithmus kein brauchbares Ergebnis lieferte, sind praktisch solche, in denen die Neuübertragungen deutlich früher als nach einer Sekunde kamen, oder, in denen der angesprochene Host unsinnige und widersprüchliche Reaktionen lieferte, etwa erst ein TCP-Rst-Paket und kurz danach ein TCP-Ack-Paket. Tabelle 6.1 zeigt die Häufigkeit, mit der Hosts aufgrund bestimmter Umstände unbrauchbar waren. Da bei einem Host mehrere Gründe gleichzeitig zutreffen können, ist die Summe aller Positionen größer, als die tatsächliche Zahl der unbrauchbaren Hosts. In elf Fällen hat die Heuristik RTT_{max} großzügiger bestimmt, als es den erkennbare Neuübertragungen zufolge richtig ist. In diesen Fällen ist die Paketverlustrate auf dem Hinweg größer als auf dem Rückweg, so dass tatsächlich mehr erkennbare als nicht-erkennbare Neuübertragungen vorlagen. In einzelnen Fällen hat die Heuristik RTT_{max} kleiner als 1 Sekunde gewählt, ohne dass erkennbare Neuübertragungen vorlagen. In diesen Fällen war die RTT-Verteilung so unregelmäßig, dass auch bei manueller Untersuchung nicht zu ermessen war, ob nicht-erkennbare Neuübertragungen vorlagen oder nicht. Auch hier liegt man also nur durch Verwerfen des Hosts auf der sicheren Seite. Immerhin sind nach dieser Untersuchung 1006 von 1064 Hosts brauchbar, also knapp 95 %.

Erst eine Rst- dann eine Ack-Antwort	9
Erst eine Ack- dann eine Rst-Antwort	24
Eine erkennbare Neuübertragung nach weniger als einer Sekunde	9
Die Heuristik liefert ein RTT_{max} kleiner als eine Sekunde	15
Eine erkennbare Neuübertragung ist kleiner als RTT_{max}	11
Keine Daten (nur Paketverluste)	9
Unbrauchbare Hosts insgesamt	58
Zahl der untersuchten Hosts	1064

Tabelle 6.1: Analyse der Host-Reaktionen und Ergebnisse der Heuristik.

Kapitel 7

Performance der Anwendungsschicht

7.1 Motivation

Für den Benutzer sind die Performance-Maße der Netzebene wie Paketlaufzeit und Paketverlustrate abstrakte Größen, die für ihn keine konkrete Bedeutung haben, auch wenn sie noch so grundlegend und klar definiert sind. Er interessiert sich statt dessen für die Performance seiner *Anwendung*, also die der zwei Stufen höher liegenden Schicht (siehe Kapitel 1.2). Da die Zahl der Anwendungen und Anwendungs-Protokolle im Internet unüberschaubar ist, werden hier einige, wenige *Klassen* von Anwendungen betrachtet (siehe Kapitel 1.3.2). Dass sich auf die vorgestellten Anwendungs-Klassen tatsächlich der größte Teil des heutigen Internet-Verkehrs verteilt, wird anhand aktueller Verkehrsbeobachtungen deutlich gemacht.

Bei den folgenden Betrachtungen kommt dem Verhalten des eingesetzten Transportprotokolls große Bedeutung zu, da es im Protokollstapel genau zwischen Netz- und Anwendungsschicht steht. Das Transport-Protokoll setzt die durch bestimmte Maße beschreibbare Performance der Netzschicht um in die Performance der Transportschicht, die sich ihrerseits durch Maße beschrieben läßt, die je nach Anwendungs-Klasse direkt für die Performance der Anwendung maßgeblich sind.

7.2 Verteilung des Internet-Verkehrs

Tabelle 7.1 zeigt die typische Verteilung des heutigen Internet-Verkehrs auf die gängigen Transport-Protokolle. Die Tabelle basiert auf den Daten, die im Internet-Verkehr der Universität zu Köln für das Jahr 1998 gemessen wurden. Es wird der Anteil am Datenvolumen angegeben. Nähere Angaben finden sich im Jahresbericht des Rechenzentrum der Universität [86]. Mit über 90 % hat TCP klar den größten Anteil am Datenverkehr.

TCP	92.87 %
UDP	5.91 %
andere (GRE, ICMP, IGMP)	1.22 %

Tabelle 7.1: Aufteilung des Internet-Verkehrs auf Transportprotokolle.

Wie sich der TCP-Verkehr auf Anwendungsprotokolle verteilt, ist in Tabelle 7.2 dargestellt. Grundlage ist wieder der Internet-Verkehr der Universität zu Köln im Jahr 1998. Mit Abstand den größten Anteil haben die Protokolle HTTP und NNTP, gefolgt von FTP. Mit dem FTP-Protokoll werden typischerweise größere Dateien in eine Richtung übertragen, so dass dieser Dienst von relativ wenigen, aber langen TCP-Verbindungen gekennzeichnet ist. Noch extremer ist das Verhältnis bei NNTP: eine Einrichtung betreibt meist einen zentralen NetNews-Server, der einige, wenige Verbindungen zu Servern anderer Einrichtungen betreibt und diese Verbindungen oft stunden- oder tagelang aufrechterhält und sehr große Datenmengen überträgt. Dabei finden sowohl lang andauernde Übertragungen in beide Richtungen statt, als auch ein gewisser Dialog zwischen den Servern. Wenn der sogenannte „Streaming Mode“ verwendet wird, werden die lang andauernden Übertragungen praktisch gar nicht unterbrochen. Die Verteilung auf die Endbenutzer findet typischerweise nur innerhalb der Einrichtung statt. Anders verhält sich HTTP. Dabei baut jeder einzelne Benutzer viele einzelne Verbindungen zu vielen, weit verteilten Servern auf, wobei pro Verbindung meist nur vergleichsweise kleine Datenmengen übertragen werden. In einigen Fällen allerdings wird auch HTTP ähnlich wie FTP zum Transport großer Dateien verwendet.

HTTP (WWW)	32.24 %
NNTP (NetNews)	32.21 %
FTP	12.59 %
X11 (remote GUI)	1.76 %
SMTP (E-Mail)	1.67 %
HTTPS (WWW, verschlüsselt)	0.21 %
telnet (remote console)	0.16 %
Rest	19.16 %

Tabelle 7.2: Aufteilung des TCP-Verkehrs auf Anwendungsprotokolle.

7.3 TCP-basierte Dateiübertragung

Eine der wichtigsten Anwendungen von TCP ist es, im Internet große Dateien von einem Host zu einem anderen zu übertragen. Die Performance der Anwendung besteht dann im wesentlichen aus der *Datenübertragungsrate*, gemessen etwa in Byte pro Sekunde.

Obwohl die ursprüngliche Spezifikation des Protokolls TCP [67] mit ihren 85 Seiten schon eine gewissen Komplexität hat, sind im Laufe der vergangenen Jahre weitere Anforderungen hinzugekommen [40, 78, 46, 2], die die Komplexität weiter verstärken. Ein wesentlichen Teil dieser Komplexität dient dabei der Verfeinerung einer der herausragendsten Eigenschaften von TCP: solange der Sender schnell genug Daten liefern und der Empfänger sie verarbeiten kann, versucht TCP beständig, die Datenübertragungsrate auf das größtmögliche Niveau zu bringen, das das zugrundeliegende Netz bieten kann, ohne jedoch das Netz durch das Zusammenwirken vieler TCP-Verbindungen zusammenbrechen zu lassen. Dabei kommen eine Reihe von Mechanismen zum Einsatz:

Slow Start sorgt dafür, dass eine neue TCP-Verbindung mit niedriger Datenübertragungsrate beginnt, diese dann aber rasch erhöht, um das netztechnisch machbare Maximum zu erreichen.

Congestion Avoidance erkennt Anzeichen von Netzüberlastungen — typischerweise Paketverluste — und reduziert daraufhin die Datenübertragungsrate des Senders, damit die Netzüberlastung abgebaut werden kann. Anschließend wird die Datenübertragungsrate vorsichtig wieder erhöht, um sich auf eine möglicherweise wieder verbesserte Netzsituationen einzustellen.

Fast Retransmit verbessert das Verhalten bei starken Paketverlusten und erlaubt es dem Sender, sich schneller wieder auf normale Datenübertragungsraten einzustellen.

Die Spezifikationen lassen einen gewissen Spielraum bei der Implementierung zu und es werden ständig weitere Verbesserungen vorgeschlagen, so dass sich das Verhalten der gängigen TCP-Implementierungen immer wieder voneinander unterscheidet. Welche Implementierung nun unter bestimmten Gesichtspunkten Vorteile gegenüber anderen hat und welche weiteren Protokoll-Ergänzungen die Performance von TCP weiter verbessern könnten, ist Gegenstand laufender Untersuchungen [22, 48].

Vor diesem Hintergrund erscheint es nicht möglich oder sinnvoll, eine exakte, analytische Beschreibung der Performance von TCP im allgemeinen oder einer speziellen Implementierung zu finden. Verschiedenen Forschergruppen ist es jedoch mittlerweile gelungen, mit Hilfe von vereinfachenden

Modellen eine analytische Beschreibung der TCP-Performance in Abhängigkeit der Netz-Performance zu finden, die zumindest unter bestimmten Bedingungen zutrifft und eine gute Näherung an die tatsächlich messbare Performance bietet.

7.3.1 Primitives Modell

Für sehr niedrige Paketverlustraten p lässt sich leicht ein einfaches Modell finden: man ignoriert die Paketverluste völlig, $p = 0$. Dann wird TCP nach dem Slow Start bald bei der maximalen Fenstergröße W_{max} ankommen. Das „Fenster“ gibt bei TCP vor, wie viele Daten, deren Empfang noch nicht bestätigt wurde, der Sender maximal ins Netz abgeben darf. Die effektive Fenstergröße ist daher direkt maßgeblich für die Datenübertragungsrate, wird allerdings durch die erwähnten Mechanismen laufend dynamisch begrenzt. Bei einem festen Fenster von W_{max} würde man einen gleichmäßigen Durchsatz erhalten:

$$\lambda_{\mathrm{TCP}} = \frac{W_{\max}}{RTT} \tag{7.1}$$

mit
λ_{TCP} : TCP-Datenübertragungsrate
RTT : Paketlaufzeit (Round Trip Time)

Dieses Modell berücksichtigt keinen der genannten Mechanismen, mit denen TCP seine Datenübertragungsrate reguliert. Daher ist es praktisch nur zu gebrauchen, um eine obere Schranke für die unter bestimmten Bedingungen maximal mögliche Datenübertragungsrate zu bestimmen.

7.3.2 Modell nach Mathis et al.

Ein erstes interessantes Ergebnis brachte die Arbeit von Mathis et al. [47]. Anhand eines einfachen Modells entwickeln sie folgende Formel für die TCP-Datenübertragungsrate im „stationären Fall“:

$$\lambda_{\mathrm{TCP}} \leq \left(\frac{MSS}{RTT}\right) \frac{1}{\sqrt{p}} \tag{7.2}$$

mit
MSS : Paketgröße (Maximum Segment Size)
p : Paketverlustrate

Das zugrundliegende Modell hat allerdings einige Einschränkungen:

- Es berücksichtigt unter den verschiedenen Mechanismen zur Regulierung der Datenübertragungsrate nur Congestion Avoidance, nicht aber Slow Start. Generell bleiben alle Vorgänge zu Beginn einer TCP-Verbindung unberücksichtigt, so dass das Modell nur sehr lange TCP-Verbindungen sinnvoll beschreiben kann, bei denen Verbindungsaufbau und Slow Start vernachlässigt werden können.

- Des weiteren werden keine Timeouts berücksichtigt. Zu einem Timeout kommt es, wenn alle innerhalb eines Fensters vom Sender zum Empfänger geschickten Pakete verloren gehen. Der Empfänger hat dann keine Informationen darüber, dass senderseitig noch Pakete ausstehen, und liefert dementsprechend keine Rückmeldung. Daher kann der Sender auch nicht sofort auf den Paketverlust reagieren, sondern erst, wenn er nach dem Ablauf eine bestimmten Zeitspanne (dem Retransmit-Timer) noch keine Antwort vom Empfänger bekommen hat. Timeouts treten vor allem dann auf, wenn die Paketverlustrate p groß ist. Das Modell liefert dann zu optimistische Werte für die TCP-Datenübertragungsrate.

- Außerdem geht das Modell von einem beliebig großen TCP-Fenster aus. Aus protokolltechnischen und praktischen Gründen gibt es jedoch immer auch eine statische Obergrenze für das Fenster, nicht zuletzt weil ein Betriebssystem für jede TCP-Verbindung Speicher in dieser Größe reservieren muss. Da das Modell diese Grenze ignoriert, gibt es auch für kleine Paketverlustraten p zu optimistische Werte an. Geht p gegen Null, so gehen Fenstergröße und die Datenübertragungsrate gegen Unendlich.

Trotz dieser Einschränkungen war dieses Modell ein Fortschritt, weil es für lange TCP-Verbindungen bei mittleren Paketverlustraten erstmalig Aussagen über die zu erwartende Datenübertragungsrate machen konnte. In der gleichen Arbeit wird Gleichung 7.2 noch um empirisch ermittelte Korrekturparameter erweitert und erreicht dadurch eine bessere Annäherung an gemessene Ergebnisse.

7.3.3 Modell nach Padhye et al.

Ein feineres Modell, das auch Timeouts und die maximale Fenstergröße berücksichtigt, wurde von Padhye et al. vorgestellt [58]. Auch dieses Modell macht einige Abstraktionen und vereinfachende Annahmen. Beispielsweise wird eine TCP-Verbindung willkürlich in „Runden“ aufgeteilt, was keine Entsprechung in der Realität hat. Und es wird angenommen, dass Paketverluste innerhalb einer Runde stark, darüber hinaus aber gar nicht korreliert sind. Tatsächlich zeigen Untersuchungen, dass die Verlustwahrscheinlichkeit von aufeinanderfolgenden Paket sehr viel stärker korreliert ist, als die zeitlich weiter auseinanderliegender Pakete [62, 61]. Trotzdem ist obige Annahme natürlich eine starke Vereinfachung der Realität.

In einer neueren Arbeit [57] wird das Modell weiter verfeinert und als Markov-Kette dargestellt. Allerdings wird dabei nur vergleichsweise wenig Genauigkeit hinzugewonnen und das Markov-Modell hat den Nachteil, sich nur numerisch, nicht aber analytisch lösen zu lassen.

In der gleichen Arbeit unterscheiden Padhye et al. drei verschiedene Definitionen der Datenübertragungsrate:

Send Rate beschreibt die Rate, mit der der Sender Daten ins Netz abgibt.

Throughput bezieht sich nur auf die beim Empfänger ankommenden Daten. Throughput unterscheidet sich von Send Rate also durch die Paketverluste im Netz.

Goodput schließlich beschränkt sich auf die nicht-doppelt beim Empfänger ankommenden Daten. Goodput unterscheidet sich von Throughput durch Doppelübertragungen, die bei TCP nie vollständig vermieden werden können.

Wurde in der ursprünglichen Arbeit nur die Send Rate bestimmt, so wird mit der neueren Arbeit eine analytische Berechnung für Throughput TP nachgeliefert:

$$TP = \begin{cases} \dfrac{MSS\left(\frac{1-p}{p}+1/2\,w(p)+Q(p,w(p))\right)}{RTT\,(w(p)+1)+\frac{Q(p,w(p))G(p)T_0}{1-p}} & \text{falls} \quad w(p) < w_{max} \\ \dfrac{MSS\left(\frac{1-p}{p}+1/2\,w_{max}+Q(p,w_{max})\right)}{RTT\left(\frac{w_{max}}{4}+\frac{1-p}{p\,w_{max}}+2\right)+\frac{Q(p,w_{max})G(p)T_0}{1-p}} & \text{sonst} \end{cases} \tag{7.3}$$

mit

$$\begin{aligned} w_{max} &= W_{max}/MSS \\ w(p) &= 2/3 + 2/3\,\sqrt{3\,\frac{1-p}{p}+1} \\ Q(p,w) &= \min\left(1, \frac{\left(1-(1-p)^3\right)\left(1+(1-p)^3\left(1-(1-p)^{w-3}\right)\right)}{1-(1-p)^w}\right) \\ G(p) &= 1 + p + 2\,p^2 + 4\,p^3 + 8\,p^4 + 16\,p^5 + 32\,p^6 \\ T_0: &\quad \text{Retransmission Timer} \end{aligned} \tag{7.4}$$

Den Anwender interessiert aber eigentlich nur *Goodput*, weil das letztlich die von der Transportschicht an die Anwendungsschicht abgegebene Datenübertragungsrate beschreibt. Da eine analytische Beschreibung für Goodput bislang nicht gefunden wurde, kann man die für Throughput als Näherung verwenden. Die Untersuchungen von Padhye et al. zeigen, dass bereits Send Rate und Throughput in weiten Bereichen in ähnlichen Größenordnungen liegen. Da normalerweise Doppelübertragungen im Internet weit seltener sind als Paketverluste, ist die hier getroffene Näherung durchaus akzeptabel.

Ein paar Schwächen bleiben bei diesem Modell:

- Slow Start wird nicht berücksichtigt, die Näherung gilt also auch nur für lange TCP-Verbindungen.

- Es lassen sich nur Throughput und Send Rate berechnen, nicht aber Goodput. Throughput als Näherung für Goodput liefert tendenziell zu optimistische Ergebnisse.
- Der Mechanismus Fast Retransmit bleibt unberücksichtigt. Das liefert tendenziell zu pessimistische Ergebnisse.
- Die Paketverlustrate p wird mit einer Rate an „Congestion Indication“ gleichgesetzt. Es ist unklar, ob das in der Praxis eher der bidirektionalen oder unidirektionaler Paketverlustrate entspricht[1].

Gleichung 7.3 liefert für den gesamten praxisrelevanten Bereich von Paketverlustrate und Paketlaufzeit eine gute Näherung der TCP-Performance.

7.3.4 Vergleich der Modelle

Abbildung 7.1 vergleicht die drei Modelle. Dargestellt ist die Abhängigkeit der TCP-Performance nach dem jeweiligen Modell in Abhängigkeit von der Paketverlustrate p. Für die übrigen Parameter wurden solche Werte gewählt, die in der Praxis üblich sind oder häufig auftreten: $RTT = 150\,\text{ms}$, $MSS = 1460\,\text{Byte}$, $W_{max} = 8760\,\text{Byte}$, $T_0 = 3\,\text{s}$.

Man kann das primitive Modell („simple“) und das nach Mathis et al. als Näherungen an das genauere Modell nach Padhye et al. verstehen, die in unterschiedlichen Bereichen von p gültig sind. Eine Kombination aus diesen Modellen wurde daher auch in eine früheren Veröffentlichung [31] verwendet, um die TCP-Performance über einen größeren Bereich von p zu bestimmen. Für $p = 0$ ist Gleichung 7.3 nicht definiert. Sie nähert sich dann aber Gleichung 7.1 asymptotisch.

7.3.5 Beispiel

Ein Beispiel, wie sich mit Hilfe der vorgestellten Messungen und Modelle konkrete und sinnvolle Fragen aus Sicht der Benutzer beantworten lassen, zeigt Abbildung 7.2: Die dargestellten Verteilungsfunktionen geben an, zu einem wie großen Anteil des weltweiten Internet TCP-Übertragungen mit einer bestimmten Übertragungsrate durchgeführt werden können. Dabei werden zwei verschiedene Internet-Provider sowie zwei verschiedene Tageszeiten unterschieden.

Die hier zugrundeliegenden Messungen wurden an den 5 Werktagen der Woche 37 des Jahres 1999 durchgeführt, und zwar stündlich mit je 20 TCP-SYN-Paketen zu 1000 Hosts. Zwei Tageszeiten wurden exemplarisch herausgegriffen und alle Messungen der Woche, die in der genannten Stunde stattfanden, wurden zusammengefasst.

[1] Sowohl empirische Untersuchungen als auch aus der Literatur ableitbare Vermutungen weisen darauf hin, dass hier die unidirektionle Paketverlustrate anzuwenden ist, siehe Kapitel 8.2.

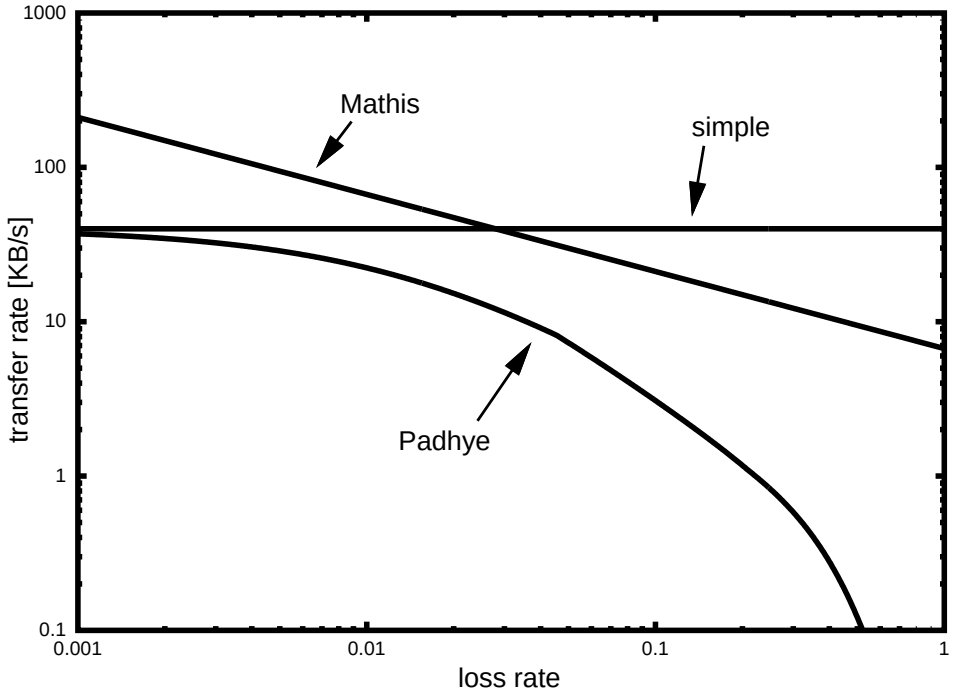

Abbildung 7.1: TCP-Datenübertragungsrate nach verschiedenen Modellen.

Provider 1 kann zu einen gewissen Teil der Internet besonders hohe Übertragungsraten bieten und übertrifft damit Provider 2.

Bei beiden Providern erkennt man, dass die Netz-Leistung am Tage, wenn viele Benutzer aktiv sind und das Netz stärker ausgelastet ist, deutlich zurückgeht. Während dieser Vorgang bei Provider 2 sehr mäßig und ausgeglichen auftritt, so ist er bei Provider 1 für große Teile des Internet dramatisch. Über Provider 1 kann man die Hälft des Internet nur mit 2.6 KB/Sekunde oder weniger erreichen.

Für den Benutzer lassen sich nun je nach Anforderung konkrete Aussagen ableiten:

Fallbeispiel 1 Der Benutzer hat eine besonders anspruchsvolle Anwendung, bei der er mit bestimmten Partnern im Internet bei sehr hoher Übertragungsrate Daten austauschen muss. Unter der Annahme, dass Provider 1 gerade zu diesen Partner die erwähnte besonders gute Konnektivität hat, ist für diesen Benutzer Provider 1 zu empfehlen.

Fallbeispiel 2 Der Benutzer muss tagsüber in vielen verschiedenen Teilen des Internet recherchieren. Er benutzt einen Wählzugang mit ISDN, der die Übertragungsrate seinerseits auf knapp 8 Kbyte/Sekunde begrenzt. Da der Wählzugang nach Online-Zeit abgerechnet wird, soll die

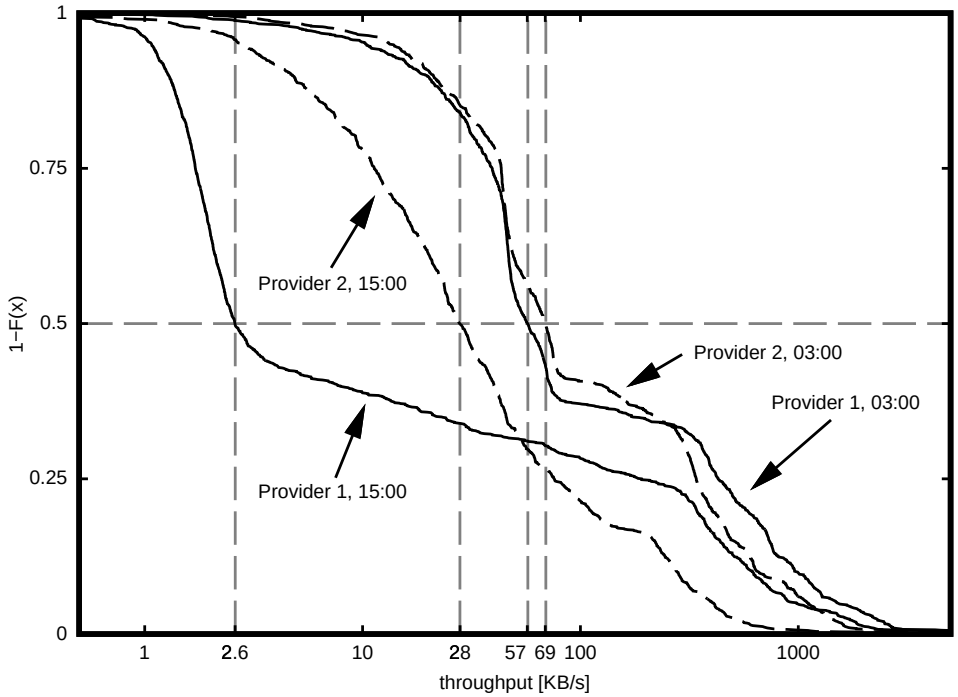

Abbildung 7.2: Verteilung der TCP-Datenübertragungsraten bei verschiedenen Providern und zu verschiedenen Tageszeiten.

so begrenzte Übertragungsrate so oft wie möglich auch erreicht werden. Provider 1 wäre hier sehr schlecht geeignet, da zu über 60 % des Internet diese Übertragungsrate nicht erreicht werden kann. Provider 2 hätte bei weniger als 18 % des Internet solche Einschränkungen und stellt sich somit wesentlich besser dar.

7.4 Echtzeitanwendungen

Echtzeitanwendungen wie IP-Telefonie, Musik- und Video-Übertragungen, Sprach- und Video-Konferenzen machen zur Zeit noch einen relativ kleinen Teil des Internet-Verkehrs aus — sie bestehen bei der in Tabelle 7.1 dargestellten Verkehrsverteilung aus einem Teil des UDP-Verkehrs und aus dem GRE-Verkehr (MBone-Tunnel). Allerdings werden ihnen für die Zukunft besonders hohe Wachstumsraten nachgesagt, so dass sich die Verhältnisse durchaus ändern können.

7.4.1 Modell

Die Anforderungen der derzeit bekannten Echtzeitanwendungen an die Performance der Netzschicht lassen sich in etwa so ausdrücken:

- Eine gewisse **Paketverlustrate** p wird akzeptiert, aber sie darf einen bestimmten Wert p_{max} nicht überschreiten, sonst wird beispielsweise die Sprachübertragung unverständlich.

$$p \overset{!}{<} p_{\max} \tag{7.5}$$

- Die **Varianz der Paketlaufzeit** σ_{RTT} darf ein bestimmtes Maß $\sigma_{RTT,\max}$ nicht überschreiten, weil sonst typische Decoder in ihrer Pufferfähigkeit überfordert werden und die „Echtzeit“ nicht mehr gegeben ist.

$$\sigma_{RTT} \overset{!}{<} \sigma_{RTT,\max} \tag{7.6}$$

- Wenn die Anwendung *interaktiv* ist wie bei Telefonaten oder Konferenzen, dann darf die **Paketlaufzeit** RTT selber einen bestimmten Wert $RTT_{\max}$ nicht überschreiten, weil sonst die Interaktivität erheblich gestört wird. Ist die netzbedingte Verzögerung in der Größenordnung normaler Sprechpausen oder darüber, so fallen sich Kommunikationspartner ohne spezielle Übung immer wieder gegenseitig ins Wort.

$$RTT \overset{!}{<} RTT_{\max} \tag{7.7}$$

Dabei sind die genannten Grenzen meist nicht scharf definiert sondern weich und lassen sich nicht allgemeingültig festlegen. Sie unterscheiden sich von Codec zu Codec und sind immer auch von der subjektiven Wahrnehmung der Benutzer abhängig.

7.4.2 Beispiel

Ein Beispiel für eine Auswertung der Messungen mit Blick auf Echtzeitanwendungen zeigt Abbildung 7.3. Hier wird dargestellt, welcher Anteil des Internet für eine Echtzeitanwendung geeignet ist und wie er sich im Tagesverlauf verhält. Es liegen die gleichen Messungen zugrunde wie in Kapitel 7.3.5. Für $p_{\max}$ wurde 5 % gewählt und für $\sigma_{RTT,\max}$ 200 ms. Dies entspricht in etwa den Anforderungen an eine Echtzeit-Übertragung des Sprachkanals einer Vorlesung oder eines Vortrags.

Es werden wieder zwei Provider im Vergleich dargestellt. Während über Provider 1 tagsüber wieder nur ein gewisser Teil des Internet mit brauchbarer Qualität erreicht wird, weniger als 50 %, kann über Provider 2 tagsüber ein deutlich größerer Teil der Internet für eine solche Echtzeitanwendung genutzt werden.

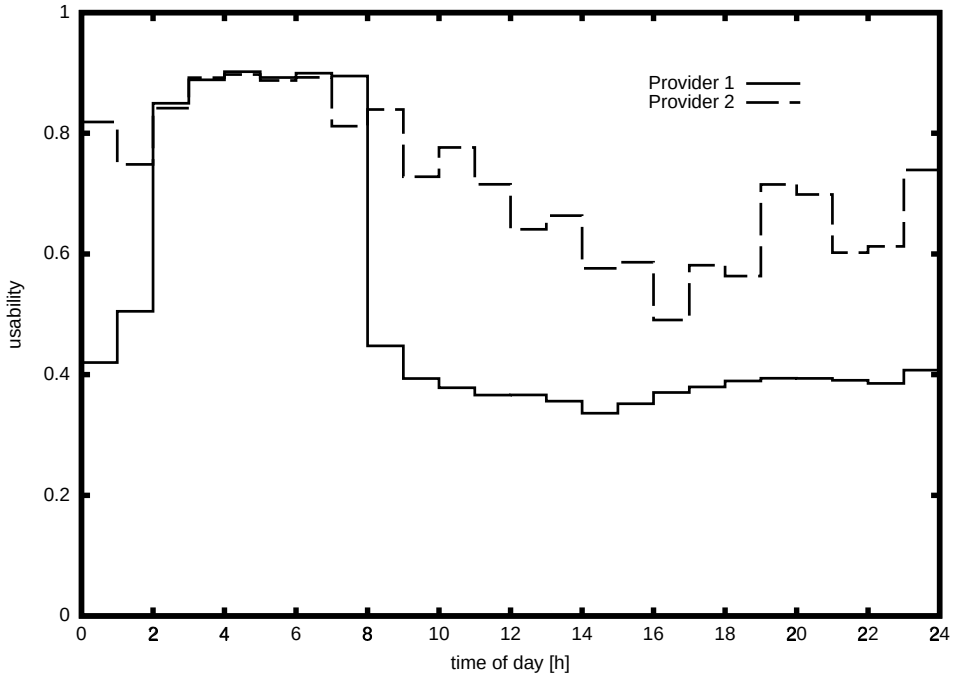

Abbildung 7.3: Tagesverlauf der Benutzbarkeit für Echtzeitanwendungen.

7.5 Konnektivität

Eine Größe der Netzschicht, die unmittelbare Bedeutung für die Anwendungsschicht hat und einem Anwender direkt verständlich ist, ist die *Konnektivität*. Mittelt man die Konnektivität über alle untersuchten Hosts, so erhält man eine Aussage darüber, welcher Anteil des Internet über einen bestimmten Provider zu einer bestimmten Zeit erreichbar war und welcher nicht.

Abbildung 7.4 zeigt die TCP-Syn-Intervall-Normalkonnektivität für zwei verschiedene Provider und die UDP-Echo-Intervall-Normalkonnektivität für den ersten Provider im Laufe einer Woche. Grundlage sind die stündlichen Messungen nach Kapitel 5 und 6 zu je 1000 Hosts. Die UDP-Echo-basierte Konnektivität ist hier viel niedriger als die TCP-Syn-basierte, weil viele Hosts auf die UDP-Echo-Pakete überhaupt nicht antworten und dieser Umstand hier nicht kompensiert wurde.

In der Abbildung lassen sich zwei Einbrüche in der Konnektivität von Provider 1 erkennen und zwar am Sonntag und am Montag Vormittag. Diese Konnektivitäts-Einbrüche werden sowohl nach den UDP-Echo-basierten als auch nach den TCP-Syn-basierten Messungen deutlich. Sie zeigen kurzzeitige Störungen des weltweiten Routings dieses Providers an — vermutlich eine

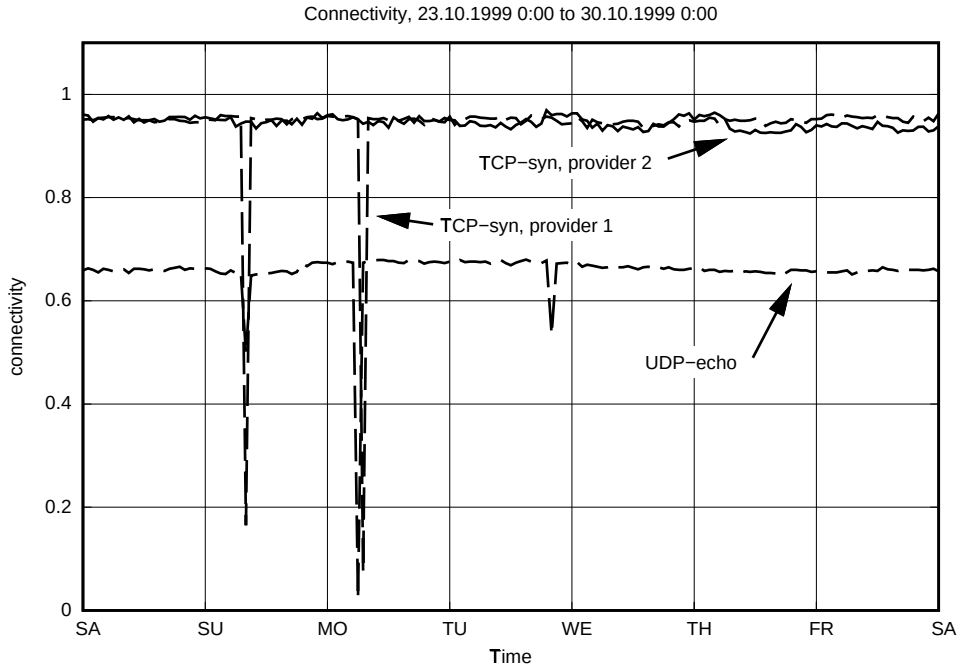

Abbildung 7.4: Verlauf der Konnektivität zwei verschiedener Provider über eine Woche.

Folge einer grundlegenden Topologieänderung im Netz dieses Providers, die genau in der Vorwoche stattgefunden hatte. Des weiteren ist ab Donnerstag Vormittag ein leichter Einbruch der Konnektivität von Provider 2 erkennen. Dieser Einbruch dauerte mehrere Tage, bis sich die Konnektivität wieder normalisierte. Als Ursache lässt sich eine Störung des Routings dieses Providers zu einem relativ kleinen Teil des weltweiten Internets vermuten, die nicht schwerwiegend genug war, um das Personal des Providers sofort zu alarmieren, so dass das Problem erst nach einigen Tagen erkannt und behoben wurde.

Obgleich es sich nur um ein Nebenprodukt der vorliegenden Arbeit handelt, hat sich eine derartige Darstellung der Messergebnisse im Netzbetrieb des Regionalen Rechenzentrum der Universität zu Köln bereits mehr als einmal als sehr nützlich erwiesen, als es um das Erkennen von Fehlfunktionen im internationalen Routing ging und um die Abschätzung ihrer Dauer.

Kapitel 8

Empirische Validierung

8.1 Motivation

In Kapitel 1 wird ein allgemeines Modell des Internet vorgestellt und in 1.3.1 begründet, warum quantitative Messungen der Internet-Konnektivität am sinnvollsten in der *Netzschicht* stattfinden. In Kapitel 4 wird gezeigt, wie man eine Liste vieler Hosts im Internet gewinnen kann, die das ganze Internet repräsentieren können. In Kapitel 5 und 6 werden Verfahren vorgestellt, die in Kapitel 3 vorgestellten Performance-Maße der Netzschicht effektiv zu messen. Kapitel 7 schließlich zeigt, wie man von der Performance der Netzschicht auf die Performance von Internet-Anwendungen schließen kann und nennt Beispiele für konkreten Fragen der Benutzer, die sich mit den vorgestellten Methoden beantworten lassen.

Im folgenden Kapitel soll nun überprüft werden, ob die aus abstrakten Performance-Werten und analytischen Formeln hergeleitete Performance der Anwendungsschicht mit tatsächlich beobachtbaren Anwendungen übereinstimmt, ob also die in den vorangegangenen Kapiteln gemachten Annahmen, Abstraktionen und Vereinfachungen hinreichend realistisch sind.

8.2 TCP-basierte Dateiübertragung

Die Anwendungsklasse „TCP-basierte Dateiübertragung“ wird in Kapitel 7.3 beschrieben. Gleichung 7.3 beschreibt anhand eines guten analytischen Modells (nach Padhye et al.) die TCP-Datenübertragungsrate in Abhängigkeit von den Performance-Werten der Netzschicht. Basierend auf Messungen entsprechend der Auswahl- und Messverfahren nach Kapitel 4 und 6 lässt sich damit die zu erwartende TCP-Datenübertragungsrate abschätzen. Diese wird im folgenden als **geschätzte** („estimated“) Datenübertragungsrate bezeichnet.

Die geschätzte Datenübertragungsrate sollte sich nun mit solchen Datenübertragungsraten vergleichen lassen, die von konkreten Anwendungen in

der Praxis erzielt werden, im folgenden als **beobachtete** („observed“) Datenübertragungsrate bezeichnet. Dabei sind allerdings folgende Einschränkungen zu beachten:

- Die Anwendung selber kann die Datenübertragungsrate beschränken, zum Beispiel indem der Sender nicht immer sofort Daten liefert oder der Empfänger sie nicht immer sofort abnimmt. Geeignet für einen Vergleich sind daher nur Anwendungen, bei denen davon ausgegangen werden kann, dass der Sender immer Daten liefert und der Empfänger sie umgehend abnimmt. Nicht geeignet sind daher beispielsweise NNTP-Verbindungen, bei denen die Sende-Richtung während einer TCP-Verbindung gelegentlich wechselt und in der sich Datenübertragungsphasen immer wieder mit Dialog-Phasen abwechseln. Gut geeignet sind hingegen Dateiübertragungen mit FTP oder HTTP, da hier in einer Verbindung genau eine Datei kontinuierlich gesendet und empfangen wird.

- Die Leistungsfähigkeit des Senders oder des Empfängers kann unter der des Netzes liegen und dadurch die Übertragungsrate begrenzen. Das kann beispielsweise bei stark ausgelasteten Servern der Fall sein, die viele Clients bedienen müssen. Oder bei Clients, deren Hardware (etwa CPU oder Massenspeicher) nur eine geringe Leistungsfähigkeit hat.

- Es können Engpässe in solchen Teilen des Netzes auftreten, die eigentlich nicht untersucht werden sollen. Zum Beispiel dann, wenn die Konnektivität eines Providers untersucht werden soll, die Clients aber nur über langsame Wählverbindungen mit dem Provider verbunden sind.

Zur Beobachtung einer größeren Menge TCP-Datenübertragungen wurde der zentrale Proxy-Server der Universität zu Köln herangezogen. Er wird von einer großen Zahl Benutzer aus der ganzen Universität für HTTP- und FTP-Verbindungen benutzt, weil er einen Zwischenspeicher für häufig angefragte Antworten bietet und dadurch in vielen Fällen die Zugriffszeit verkürzt und insgesamt Netz-Bandbreite spart. Man spricht dabei von einem **Caching Proxy**. In seinen Protokoll-Dateien wird für jede Verbindung vermerkt, wie groß die Übertragung in Byte war, wie lang sie in Millisekunden gedauert hat, in welchem Bereich der Universität der Client liegt und ob die Verbindung direkt ins Internet aufgebaut wurde oder aus dem Zwischenspeicher („Cache“) bedient werden konnte.

Für den folgenden Vergleich wurden alle Verbindungen aus den Proxy-Protokoll-Dateien herausgesucht, die folgende Kriterien erfüllen:

- Es wurden mehr als 50 KB Daten übertragen. Diese Bedingung schließt

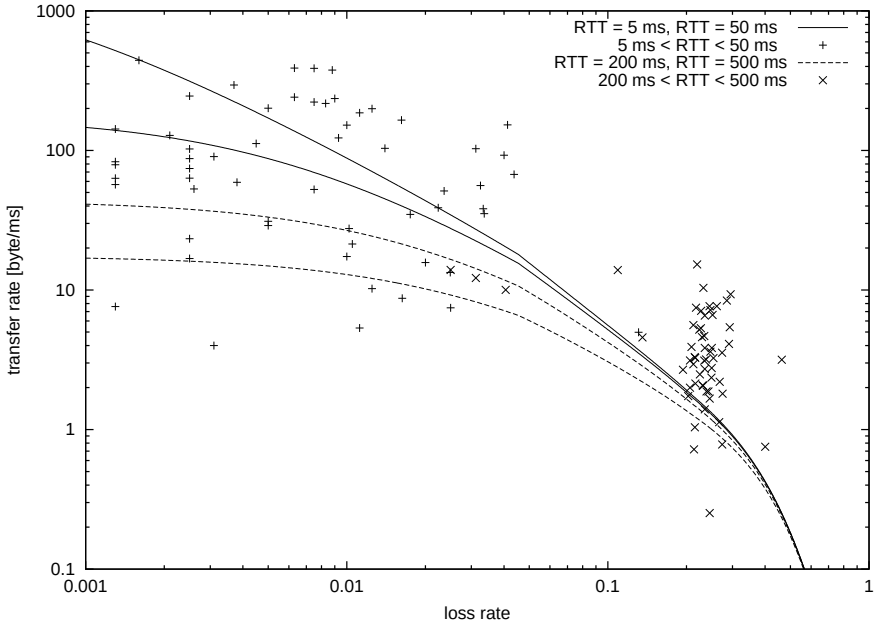

Abbildung 8.1: Abhängigkeit der TCP-Transferrate von der Paketverlustrate im Modell ($p = p_b$) und nach Messungen.

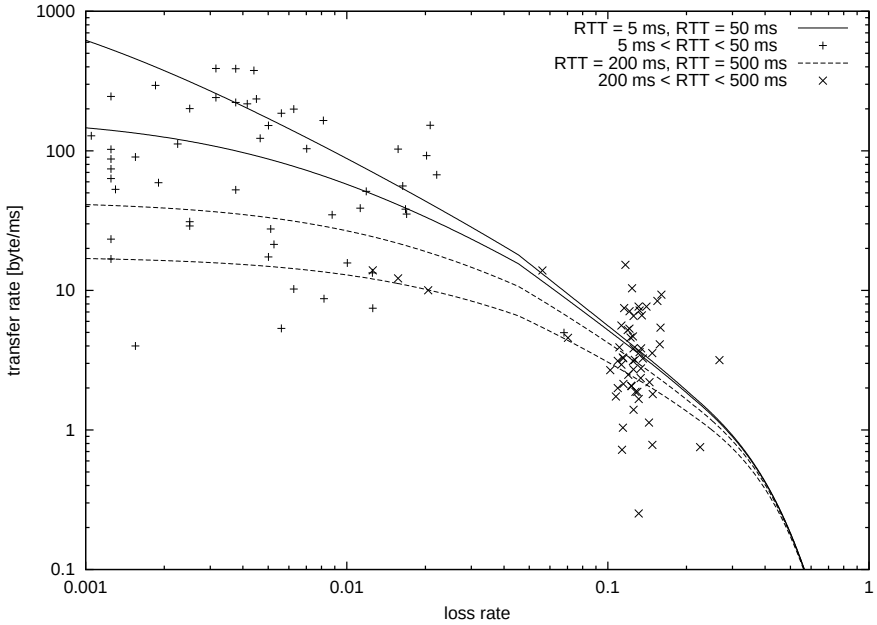

Abbildung 8.2: Abhängigkeit der TCP-Transferrate von der Paketverlustrate im Modell ($p = p_u$) und nach Messungen.

zu kurze Verbindungen aus, bei denen das verwendete Modell nicht mehr zutrifft, weil der Verbindungsaufbau zu sehr ins Gewicht fällt.

- Die Verbindung wurde ins Internet aufgebaut.
- Der Client liegt in einem Subnetz, von dem bekannt ist, dass darin keine Wählzugange oder besonders langsame Client-Maschinen liegen.

Die Verbindungen stammen aus dem gleichen Zeitraum wie die Messungen für die Schätzungen: die 5 Werktage der Woche 37 des Jahres 1999. Die Messungen wurden zu der Zeit stündlich mit je 20 TCP-SYN-Paketen zu 1000 Hosts durchgeführt. Es wurden grob drei Tageszeiten unterschieden, die erfahrungsgemäß in sich eine relativ gleichmäßige, zu den anderen aber unterschiedliche Netzsituation aufweisen: 0:00 bis 8:00 („Nacht“), 8:00 bis 16:00 („Büro-Arbeit“) und 16:00 bis 24:00 („studentische Heim-Arbeit“).

Abbildungen 8.1 und 8.2 zeigen die Abhängigkeit der TCP-Datenübertragungsrate des Modells und der beobachteten Verbindungen in Abhängigkeit von der gemessenen Paketverlustrate. In Kapitel 7.3.3 konnte nicht geklärt werden, ob als Paketverlustrate die unidirektionale oder bidirektionale Paketverlustrate eingesetzt werden muss. Abbildung 8.1 zeigt daher eine Darstellung mit bidirektionaler und Abbildung 8.2 eine mit unidirektionaler Paketverlustrate. Das Modell ist für vier Werte von RTT eingetragen, die beobachteten Übertragungsraten für die zwei angegebenen Bereiche der gemessenen Paketlaufzeit.

Mit den vorgestellten Messungen lässt sich leider nur die bidirektionale Paketverlustrate p_b messen. In der Annahme, dass die unidirektionale Paketverlustrate p_u symmetrisch, also in Hin- und Rückrichtung gleich ist, lässt sie sich wie folgt berechnen:

$$p_u = 1 - \sqrt{1 - p_b} \tag{8.1}$$

Die Annahme symmetrischer Paketverlustraten mag für den „typischen“ Fall zutreffen, wird in der Praxis aber oft widerlegt [62]. Wie oft und wie gut diese Annahme zutrifft, ist zunächst einmal also offen.

Einen direkter Vergleich zwischen geschätzter und beobachteter Übertragungsrate ist in den Abbildungen 8.5 und 8.6 dargestellt. Auf der x-Achse ist die geschätzte und auf der y-Achse die beobachtete Übertragungsrate aufgetragen. Zusätzlich ist die Gerade $x = y$ eingetragen, die den Idealfall darstellt: wenn geschätzte und beobachtete Übertragungsrate genau übereinstimmen, wird die entsprechende Verbindung durch einen Punkt genau auf dieser Geraden gekennzeichnet. In den Fällen, in denen mehrere beobachtete Verbindungen zum gleichen Host und zur gleichen „Tageszeit“ vorliegen, wird ein Fehlerbalken eingetragen, in dem die niedrigste, mittlere und höchste beobachtete Übertragungsrate markiert ist. An den langen Fehlerbalken lässt sich erkennen, dass die zu beobachtende Übertragungsrate stark schwankt: Zur gleichen Tageszeit liefert der gleiche Internet-Host

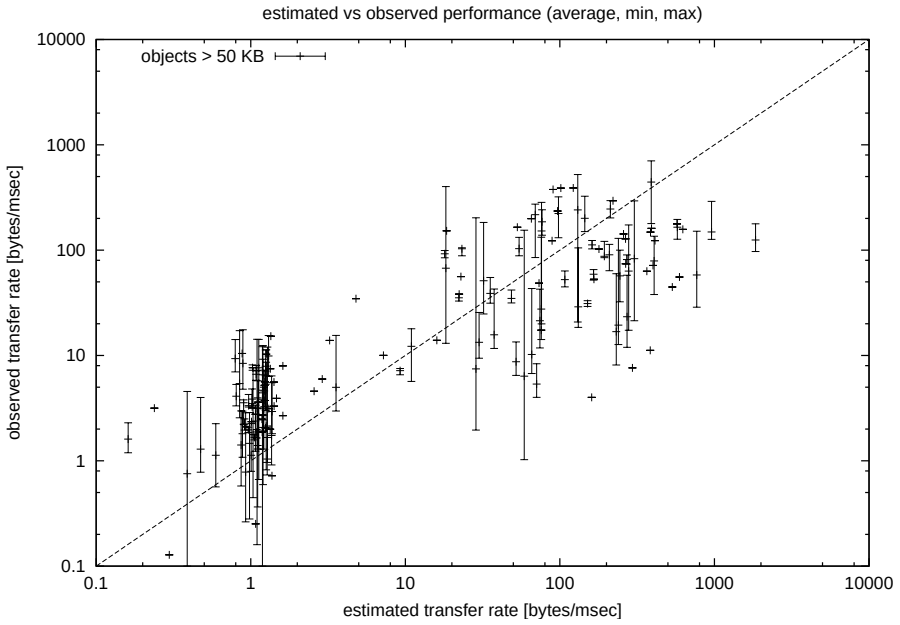

Abbildung 8.3: Vergleich der beobachteten TCP-Transferrate mit der gemäß Messung und Modell ($p = p_b$) geschätzten.

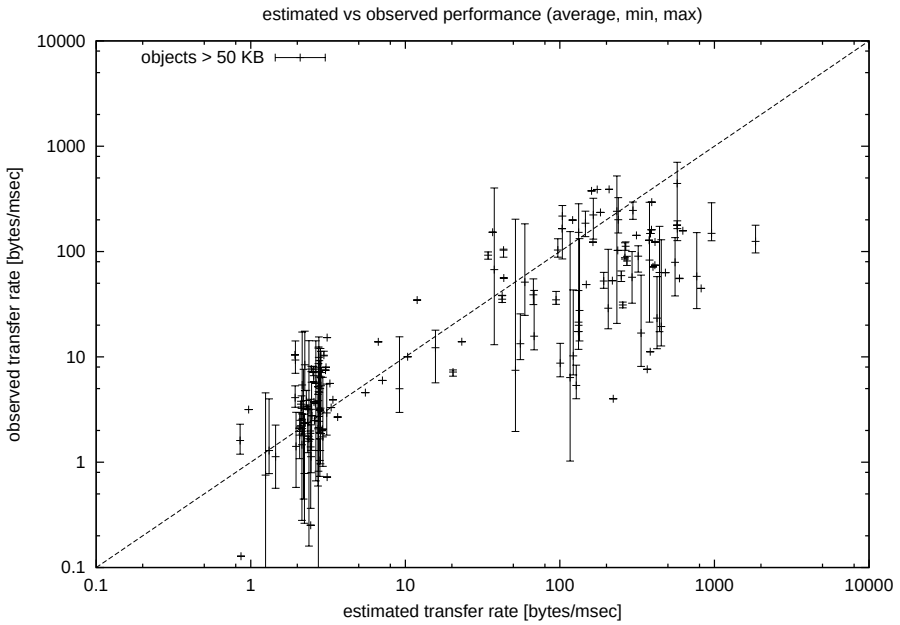

Abbildung 8.4: Vergleich der beobachteten TCP-Transferrate mit der gemäß Messung und Modell ($p = p_u$) geschätzten.

mal deutlich bessere und mal deutlich schlechtere Übertragungsraten. Diese verbessert sich auch dann nicht oder nur unwesentlich, wenn man die Tageszeit mit feinerer Granularität auflöst, beispielsweise auf die Stunde genau. Außerdem führt eine feinere Auflösung der Tageszeit dazu, dass pro Host weniger Messwerte vorliegen, so dass die Ergebnisse für p und RTT wesentlich gröber werden. Trotz dieser Einschränkungen und Schwankungen lässt sich durchaus ein Zusammenhang zwischen geschätzter und beobachteter Übertragungsrate erkennen.

Um diesen Zusammenhang noch klarer darzustellen, wurde für jede beobachtete Verbindung der Quotient aus beobachteter und geschätzter Datenübertragungsrate berechnet. Die Verteilungsfunktion der Ergebnisse ist in Abbildung 8.5 und 8.6 dargestellt. Idealerweise würde man hier eine Stufenfunktion mit einer Stufe bei $x = 1$ sehen. Um einen Eindruck zu bekommen, was realistischerweise erwartet werden kann, wird zum Vergleich die gleiche Verteilungsfunktionen für zwei andere Schätzmethoden eingetragen:

Measurement Die für eine bestimmte Tageszeit und einen bestimmten Host beobachtete Übertragungsrate wird selbst als Schätzung verwendet. Die sich ergebende Verteilungsfunktion stellt gewissermaßen den *günstigsten* Fall dar. Abweichungen von der Stufenfunktion ergeben sich nur aus den unvermeidlichen Schwankungen von Verbindung zu Verbindung.

Global Average Hier wird „globale“ der Mittelwert über alle beteiligten Hosts und alle Tageszeiten als triviale Schätzung verwendet. Daraus resultiert der *ungünstigste* Fall, der keinerlei Unterschiede zwischen den weit verteilten Hosts im Internet und den unterschiedlichen Netzsituationen zu verschiedenen Tageszeiten macht.

Die Abbildungen zeigen, dass sich die auf Messungen beruhende Schätzung deutlich von der trivialen Schätzung abhebt. Für $p = p_u$ wird eine sehr gute Annäherung an den günstigsten Fall erreicht, für $p = p_b$ weicht der Median jedoch deutlich vom Erwartungswert ab. Daraus kann lässt sich nun eine Antwort auf die in Kapitel 7.3.3 offen gebliebene Frage ableiten: das Modell basiert im wesentlichen auf der *unidirektionalen* Paketverlustrate. Dies deckt sich mit anderen Beobachtungen [61], die für TCP eine deutlich stärkere Abhängigkeit von der Paketverlustrate im Hin- als im Rückweg ausmachen.

Tabelle 8.1 führt noch einmal auf, wie viele Verbindungen mit einer gewissen Toleranz im nach den verschiedenen Methoden geschätzten Bereich liegen. Auch hier wird ein gutes Abschneiden der Messungen mit dem auf $p = p_u$ basierenden Modell deutlich: In 49.8 % der Fälle liegt darauf aufbauende Schätzung um weniger als den Faktor 2 daneben und in 93.1% der Fälle trifft sie die Größenordnung mit dem Faktor 10. Auch die ideale Schätzung

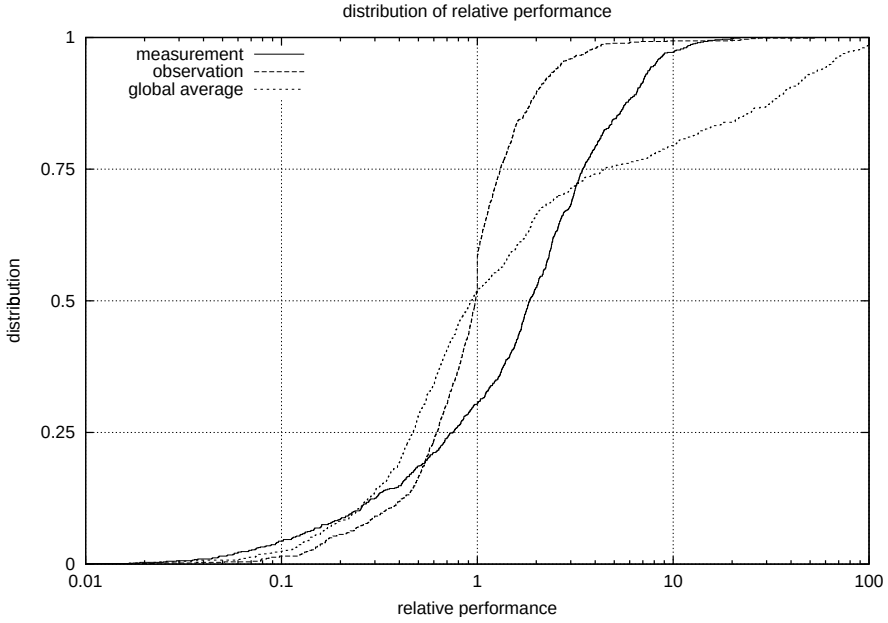

Abbildung 8.5: Verteilungsfunktion des Quotienten beobachtete / geschätzte TCP-Transferrate ($p = p_b$).

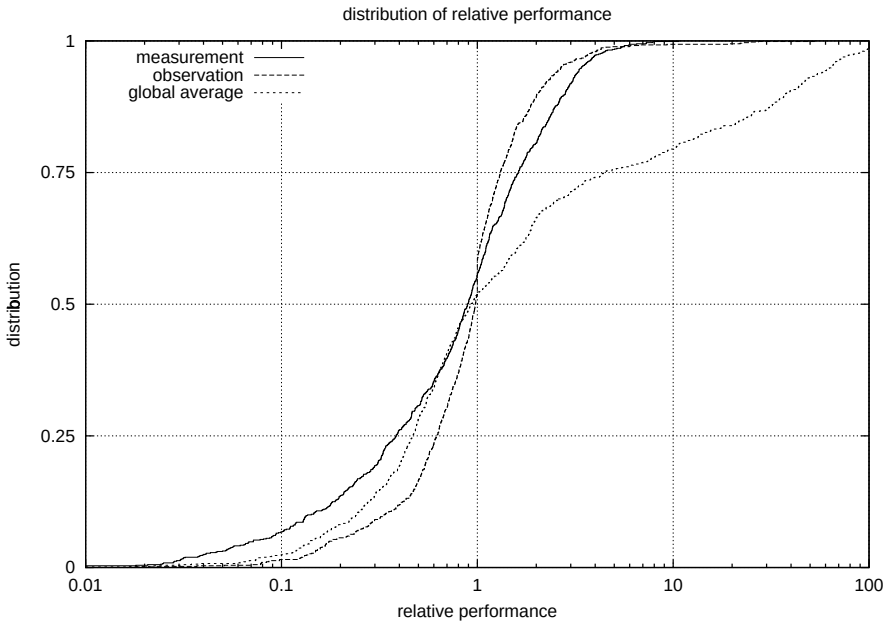

Abbildung 8.6: Verteilungsfunktion des Quotienten beobachtete / geschätzte TCP-Transferrate ($p = p_u$).

Schätzmethode	*Median* T_o/T_e	*Anteil* $\frac{1}{2} < T_o/T_e < 2$	*Anteil* $\frac{1}{10} < T_o/T_e < 10$
Modell mit $p = p_b$	1.854	34.0 %	92.8 %
Modell mit $p = p_u$	0.892	49.8 %	93.1 %
Beobachtung, pro Host gemittelt	0.978	72.9 %	97.7 %
Beobachtung, global gemittelt	0.924	38.3 %	77.0 %

Tabelle 8.1: Genauigkeit verschiedener Schätzmethoden.

(pro Host gemittelte Beobachtung) trifft diese Bereich nur in 72.9 beziehungsweise in 97.7 % aller Fälle. Die Abweichungen, die vor allem in der Nähe des Median auftreten, lassen sich zum großen Teil dadurch erklären, dass die Abschätzung der unidirektionalen aus der bidirektionalen Paketverlustrate mit Gleichung 8.1 aufgrund der im Netz üblichen Asymmetrien nur bedingt möglich ist.

8.3 Echtzeitanwendungen

Die quantitative Bewertung der Performance von Echtzeitanwendungen stößt leider auf eine Reihe von Schwierigkeiten:

- Echtzeitanwendungen im Internet sind noch relativ neu und sie haben, wie schon in Kapitel 7.4 dargestellt, derzeit nur einen kleinen Anteil am Internet-Verkehr. Daher stehen für sie nicht in gleichem Maße Erfahrungen und Performance-Beobachtungen zu Verfügung wie für die TCP-basierte Dateiübertragung.

- Es gibt bei Echtzeitanwendungen keine Entsprechung zu dem Proxy-Server der HTTP- und FTP-Verbindungen, bei dem an zentraler Stelle Daten für durchgeführte Echtzeitanwendungen gesammelt werden. Dies erschwert die Sammlung von Daten für die beobachtete Performance erheblich.

- Ein großer Teil der Echtzeitanwendungen verwendet Multicast-Routing, das derzeit nur im Multicast-Backbone (Mbone) des Internet möglich ist. Auch wenn eine Angleichung der Multicast-Topologie an die Unicast-Topologie angestrebt wird, so handelt es sich zur Zeit doch noch um sehr unterschiedliche Strukturen, so dass Messungen mit Unicast-Paketen keine relevanten Aussagen für die Multicast-Topologie zulassen.

- Andere Echtzeitanwendungen verwenden in der Regel proprietäre, nicht-öffentliche Protokolle, was eine passive Beobachtung und Analyse des Verkehrs verhindert.

- Und schließlich bleibt das Problem, dass die „Performance“ von Echtzeitanwendungen vor allem subjektiv empfunden wird und sich schlecht quantifizieren lässt.

Ein Vergleich von echter, beobachteter Leistungsfähigkeit des Netzes für Echtzeitanwendungen mit den aus Messungen gewonnen Aussagen ist daher leider auf absehbare Zeit nicht möglich. Falls sich aber die Multicast-Topologie tatsächlich weiter der Unicast-Topologie angleicht und Multicast-Routing alltäglich wird und falls sich zunehmend solche Echtzeitanwendungen durchsetzen, die mittels der im RTP-Protokoll vorgesehenen Mechanismen ihre beobachtete Dienstgüte-Parameter mitteilen, dann ist ein solcher Vergleich in der Zukunft denkbar und könnte mit guter Effizienz durchgeführt werden.

Kapitel 9

Zusammenfassung und Ausblick

Im Rahmen der vorliegenden Arbeit wurde untersucht, wie die Qualität der Einbindung eines Internet-Zugangspunktes in das weltweite Internet mit wissenschaftlichen Methoden objektiv bewertet werden kann.

Aufgrund der in Kapitel 1 dargestellten Modelle und Überlegungen wurde diese Aufgabe in drei Teile unterteilt:

1. Objektives Erstellen einer Liste vieler Internet-Hosts die – bezogen auf die Anforderungen einer gegebenen Benutzergruppe – im wesentlichen das gesamte Internet repräsentieren.

2. Messen von Performance-Größen der Netzschicht zu allen Hosts dieser Liste.

3. Ermitteln der Performance von Internet-Anwendungen anhand der gemessenen Performance der Netzschicht.

Eine Implementierung der ersten Teilaufgabe sowie mögliche Varianten werden in Kapitel 4 vorgestellt. Anhand durchgeführter Messungen wird untersucht, wie effektiv diese Methode ist: wie viele Hosts letztlich erreicht werden und welcher Anteil des Internet-Verkehrs einer typischen Benutzergruppe von ihnen repräsentiert wird.

Das Lösen der zweiten Teilaufgabe stellte sich als die größte Herausforderung dar. Zwar gibt es eine Reihe von Werkzeugen und Projekten, die sich mit Messungen in der Netzschicht befassen, aber keins wurde bisher der Anforderung gerecht, regelmäßige Messungen zu vielen Internet-Hosts zu erlauben, ohne übermäßig Bandbreite zu beanspruchen und ohne eine spezielle Kooperation bei allen diesen Hosts vorauszusetzen.

Für diese Teilaufgabe wurden zwei Messmethoden entwickelt, die mit unterschiedlichen Pakettypen arbeiten. Mit der ersten Messmethode, die mit

UDP-Paketen arbeitet, wurden erste Erfahrungen mit groß angelegten Messungen gesammelt und die Meßmethode wurde schrittweise verbessert, siehe Kapitel 5. Als unüberwindbarer Hauptnachteil dieser Methode zeigte sich, dass zwar ein nennenswerter Teil aller Internet-Hosts für die Messungen geeignet ist, dass dieser Anteil aber stetig sinkt und für eine typische Benutzergruppe nicht einmal die Hälfte des Internets repräsentieren kann.

Die zweite Messmethode verwendet TCP-Pakete und erzielt deutliche Verbesserungen in der Erreichbarkeit von Internet-Hosts. Theoretisch ist damit eine vollständige Erreichbarkeit aller Internet-Hosts garantiert, die TCP-basierte Dienste anbieten. Es zeigte sich allerdings, dass durch verschiedene Eigenheiten in der Implementierung des TCP-Protokolls bei einigen Hosts Verfälschungen der Messergebnisse für Paketverlust und Paketlaufzeit auftreten können. Diese Verfälschungen können mit einer Heuristik eliminiert werden, führen aber dazu, dass auch mit dieser Methode nicht alle interessanten Hosts erreicht werden können. Glücklicherweise bleiben hier die unbrauchbaren Hosts aber eine kleine Minderheit. Eine Implementierung dieser Messmethode einschließlich der Heuristik wird in Kapitel 6 beschrieben und untersucht.

Kapitel 7 beschäftigt sich mit der dritten Teilaufgabe, der Ermittlung der Performance der Anwendungsschicht aus den Performance-Daten der Netzschicht. Zu diesem Zweck werden die vielfältigen Internet-Anwendungen in wenige Klassen eingeteilt, die dann getrennt untersucht werden. In der Literatur gibt es mittlerweile sehr leistungsfähige Modelle für das im Internet überwiegende Transportprotokoll TCP. Diese Modelle werden untersucht und es werden Beispiele gezeigt, wie sich mit Hilfe dieser Modelle und den durchgeführten Messungen konkrete Fragen aus Anwendersicht beantworten lassen.

In Kapitel 8 werden schließlich die auf den Messungen und Modellen beruhenden Abschätzungen der Anwendungsperformance verglichen mit realer, beobachteter Anwendungsperformance, um den Erfolg des gesamten Ansatzes zu überprüfen. Trotz großer Schwankungen und vieler Unwägbarkeiten bei der Bestimmung der Anwendungsperformance wird dabei eine sehr gute Übereinstimmung gefunden.

Abschließend lässt sich also festhalten, dass der gewählte Ansatz erfolgreich ist und dass eine Kombination der entwickelten Methoden und der vorgestellten Modelle eine gute und aussagekräftige Bewertung der Qualität von Internet-Zugangspunkten ermöglicht. Allerdings setzen einige der notwendigen Maßnahmen eine gewisse Infrastruktur voraus, zum Beispiel statische und dedizierte IP-Adressen, WWW-Server, Kontrolle der DNS- und RIPE-Datenbanken und eine hinreichend große Benutzergruppe, deren Verkehr in geeigneter Weise beobachtet werden kann. Daher werden die vorgestellten Methoden nie für den einzelnen Endanwender praktikabel sein, sondern nur für vergleichsweise große Internet-Kunden oder für Internet-Provider selber.

Im Rahmen weiterer Forschung können einzelne Methoden verbessert und verfeinert werden. Dies gilt beispielsweise für die im Rahmen der TCP-basierten Messungen angewandten Heuristik, deren Genauigkeit sich möglicherweise durch aufwändigere Verfahren noch steigern ließe.

Ferner ließen sich noch weitere Klassen von Internet-Anwendungen definieren, die zu anderen Modellen und Formeln bei der Errechnung der Anwendungsperformance aus der Netzperformance führen. Ein Beispiel dafür wäre die in Kapitel 1.3.2 schon erwähnte Klasse der Einzelverbindungs-Transaktionen. Mit geeigneten Vereinfachungen sollte sich hier ein relativ einfaches Modell finden lassen, das sowohl analytisch lösbar ist, als auch einen Großteil der praxisrelevanten Internet-Verbindungen realitätsnah beschreibt.

Schließlich bleibt die Frage interessant, ob sich Echtzeit-Anwendungen weiter entwickeln, ob sie an Bedeutung gewinnen und ob sich das Multicast-Backbone der Unicast-Topologie des Internet angleicht, so dass gegebenenfalls die Anforderungen der Echtzeit-Anwendungen an die Netzperformance konkretisiert werden können und dass möglicherweise auch für Echtzeit-Anwendungen ein Vergleich von geschätzter mit echter, beobachteter Anwendungsperformance durchgeführt werden kann.

Index

Literaturverzeichnis

[1] Advanced Network & Services IPPM Program. `http://www.advanced.org/ippm.html`.

[2] M. Allman, V. Paxson, and W. Stevens. RFC 2581: TCP congestion control, 1999.

[3] G. Almes, S. Kalidindi, and M. Zekauskas. RFC 2679: A one-way delay metric for IPPM, September 1999.

[4] G. Almes, S. Kalidindi, and M. Zekauskas. RFC 2680: A one-way packet loss metric for IPPM, September 1999.

[5] G. Almes, S. Kalidindi, and M. Zekauskas. RFC 2681: A round-trip delay metric for IPPM, September 1999.

[6] ANSI. X3T9.5 and X3.139: Fiber distributed data interface (FDDI).

[7] J. Apisdorf, K. Claffy, and R. Wilder. Oc3mon: Flexible, affordable, high performance statistics collection. In *INET'97*, June 1997. Recent development at `http://www.vbns.net/stats/flows/html/index.html`.

[8] Anselm Baird-Smith. Jigsaw performance evaluation. `http://www.w3.org/pub/WWW/Jigsaw/User/Introduction/performance.html`.

[9] F. Baker. RFC 1812: Requirements for IP Version 4 Routers, 1995.

[10] T. Berners-Lee, R. Fielding, and H. Frystyk. RFC 1945: Hypertext Transfer Protocol – HTTP/1.0, 1996.

[11] Benchmarking Methodology (bmwg) Charter. `http://www.ietf.org/html.charters/bmwg-charter.html`.

[12] Daniele Bovio. The regionalization of EARN. *Computer Networks and ISDN Systems*, 25:546–553, 1992.

[13] Daniele Bovio and Greg Lloyd. Monitoring EARN networking services. *Computer Networks and ISDN Systems*, 26:343–348, 1993.

[14] R. Braden. RFC 1122: Requirements for internet hosts – communication layers, 1989.

[15] N. Brownlee, C. Mills, and G. Ruth. RFC 2063: Traffic flow measurement: Architecture, Jan 1997.

[16] Nevil Brownlee. NeTraMet web page. `http://www.auckland.ac.nz/net/NeTraMet/`.

[17] K. Claffy, D. Siegel, and B. Woodstock. A common format for the recording and interchange of peering point utilization statistics. In *NANOG*, May 30 1996.

[18] Hewlett Packard Corp. Netperf home page. `http://www.cup.hp.com/netperf/ NetperfPage.html`.

[19] K. Csikai, editor. *Fachausdrücke der Informationsverarbeitung*. IBM Deutschland GmbH, 1985.

[20] S. Deering. RFC 1112: Host Extensions for IP Multicasting, 1989.

[21] Havard Eidnes. Subject: Re: Icmp and future testing. Note sent to IPPM mailinglist, Fri, 04 Dec 1998 19:22:05 +0100. Archived at `http://www.advanced.org/IPPM/archive.2/0612.html`.

[22] Kevin Fall and Sally Floyd. Simulation-based comparisons of tahoe, reno, and sack tcp. In *Computer Communications Review*, volume 26, pages 5–21, July 1996.

[23] Sally Floyd and Van Jacobson. Random early detection gateways for congestion avoidance. In *Transactions on Networking*, volume 1, pages 397–413. IEEE/ACM, August 1993.

[24] Sally Floyd and Van Jacobson. The synchronisation of periodic routing messages. In *Proc. SIGCOMM*, pages 33–44. ACM, September 1993.

[25] Sally Floyd and Van Jacobson. The synchronisation of periodic routing messages. In *Transactions on Networking*, volume 2, pages 122–136. IEEE/ACM, April 1994.

[26] Internet Corporation for Assigned Names and Numbers. ICANN. `http://www.icann.org/`.

[27] W. Goralski. *Introduction to ATM networking*. McGraw-Hill, Inc., 1995.

[28] Joachim Hackmann. Neue Messverfahren sollen das Internet auf Trab bringen. *Computerwoche*, Nr. 5, Feb 2000.

[29] R. Händel, M. Huber, and S. Schröder. *ATM Networks: Concepts, Protocols, Applications*. Addison Wesley, 1994.

[30] Gilbert Held. *Dictionary of Communications Technology: Terms, Definitions and Abbreviations*. John Wiley and Sons, third edition, 1998.

[31] Martin Horneffer. Methods for performance-analysis of internet access points. In *Proceedings of the TERENA Networking Conference '98*, volume 30 of *Computer Networks and ISDN Systems*, pages 1607–1615. Elsevier Science, September 1998.

[32] M. Horton and R. Adams. RFC 1035: Domain Names - implementation and specification, 1987.

[33] IEEE. Standard 802.3: Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications, 1998.

[34] IETF. Differentiated Services (diffserv) Charter. `http://www.ietf.org/html.charters/diffserv-charter.html`.

[35] Internet Engineering Task Force Home Page. `http://www.ietf.cnri.reston.va.us/`.

[36] IETF. IP Performance Metrics (ippm) Charter. `http://www.ietf.org/html.charters/ippm-charter.html`.

[37] Van Jacobson. `pathchar` — infer the characteristics of internet paths. Available via ftp from `ftp.ee.lbl.gov`.

[38] Van Jacobson. `traceroute` — print the route packets take to network host. Documentation and source available via ftp from `ftp.ee.lbl.gov`.

[39] Van Jacobson. Congestion avoidance and control. In *Proc. SIGCOMM*. ACM, August 1988.

[40] Van Jacobson, R. Braden, and D. Borman. RFC 1323: TCP extensions for high performance, 1992.

[41] Claus Kalle. Schutz der UKLAN-Subnetze. *Kompass – Mitteilungen des Zentrums für Angewandte Informatik der Universität zu Köln*, 1998. `http://www.uni-koeln.de/RRZK/kompass/79/wmwork/www/k79_13.html`.

[42] Brian Kantor and Phil Lapsley. RFC 977 : Network News Transfer Protocol, 1986.

[43] M. Lottor. RFC 1296: Internet Growth, 1992.

[44] M. Mathis. Internet Performance and IP Provider Metrics. `http://www.psc.edu/~mathis/ippm/`.

[45] M. Mathis and J. Mahdavi. Diagnosing Internet congestion with a transport layer performance tool. In *INET'96*, 1996.

[46] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow. RFC 2018: TCP selective acknowledgement options, Oct 1996.

[47] M. Mathis, J. Semke, J. Mahdavi, and Teunis Ott. The macroscopic behaviour of the TCP congestion avoidance algorithm. *Computer Communication Review*, 27(3), Jul 1997.

[48] Matthew Mathis and Jamshid Mahdavi. Forward acknowledgement: Refining tcp congestion control. In *Computer Communication Review*, volume 26. ACM, October 1996.

[49] Warren Matthews and Les Cottrell. Internet end-to-end performance monitoring for the high energy nuclear and particle physics community. In *The First Passive and Active Measurement Workshop — PAM 2000*, April 2000. ISBN 0-909007-20-9, `http://pam2000.cs.waikato.ac.nz/final_program.htm`.

[50] Steve McCanne, Craig Leres, and Van Jacobson. `libpcap` — packet capture library. Documentation and source available via ftp from `ftp.ee.lbl.gov`.

[51] Steve McCanne, Craig Leres, and Van Jacobson. `tcpdump` — dump traffic on a network. Documentation and source available via ftp from `ftp.ee.lbl.gov`.

[52] René Meissner. Digital auf die Datenautobahn. *c't*, pages 118–120, Mar 1996. Verlag Heinz Heise GmbH & Co KG.

[53] René Meissner. Wege in den Stau. *c't*, pages 124–127, Jan 1996. Verlag Heinz Heise GmbH & Co KG.

[54] Tracie Monk and K. Claffy. A Survey of Internet Statistics / Metrics Activities. `http://www.tomco.net/~tmonk/metrics.htm`.

[55] David R. Musser and Alexander Stepanov. *Stl Tutorial & Reference Guide: C++ Programming With the Standard Template Library*. Professional Computing Series. Addison-Wesley, 1996.

[56] Mike Muuss and Terry Slattery. ttcp — test TCP and UDP performance. `ftp://ftp.sgi.com/sgi/src/ttcp/`.

[57] Jitendra Padhye, Victor Firoiu, and Don Towsley. A stochastic model of tcp reno congestion avoidance and control. Technical report, Dept. of Computer Science, University of Massachusetts, 1999.

[58] Jitendra Padhye, Victor Firoiu, Don Towsley, and Jim Kurose. Modeling TCP throughput: A simple model and its empirical validation. In *Proc. SIGCOMM*. ACM, September 1998.

[59] V. Paxson, G. Almes, J. Mahdavi, and M. Mathis. RFC 2330: Framework for IP performance metrics, February 1998.

[60] V. Paxson and J. Mahdavi. RFC 2678: IPPM metrics for measuring connectivity, September 1999.

[61] Vern Paxson. End-to-end internet packet dynamics. In *Proc. SIGCOMM*. ACM, September 1997.

[62] Vern Paxson. *Measurement and Analysis of End-to-End Internet Dynamics*. PhD thesis, University of California, Berkeley, April 1997.

[63] Vern Paxson and Sally Floyd. Why we don't know how to simulate the internet. In *Proc. Winter Simulation Conference*, December 1997.

[64] Jon Postel. RFC 768: User Datagram Protocol, 1980.

[65] Jon Postel. RFC 791: Internet Protocol, 1981.

[66] Jon Postel. RFC 792: Internet Control Message Protocol, 1981.

[67] Jon Postel. RFC 793: Transmission Control Protocol, 1981.

[68] Jon Postel. RFC 821: Simple Mail Transfer Protocol, 1982.

[69] John S. Quarterman, Smoot Carl-Mitchell, and Gretchen Phillips. MIDS Internet Weather Report. `http://www3.mids.org/weather/`.

[70] J. Reynolds and J. Postel. RFC 1700: Assigned numbers, 1994.

[71] Réseaux IP Européens (RIPE). `http://www.ripe.net/info/ripe/ripe.html`.

[72] Marshall T. Rose. *The open book: a practical perspective on OSI*. Prentice-Hall, 1990.

[73] Dheeraj Sanghi, Olafur Gudmundsson, and Ashok K. Agrawala. Study of network dynamics. *Computer Networks and ISDN Systems*, 26:371–378, 1993.

[74] H. Schulzrinne, S. Casner, R. Frederick, and Van Jacobson. RFC 1889: Rtp: A transport protocol for real–time applications, 1996.

[75] Jeff Sedayao and Kotaro Akita. LACHESIS: A tool for benchmarking internet service providers. In *LISA IX*, September 1995.

[76] W. Simpson. RFC 1331: The point-to-point protocol (ppp) for the transmission of multi-protocol datagrams over point-to-point links, 1992.

[77] Richard W. Stevens. *TCP/IP Illustrated*, volume 1: the protocols. Addison-Wesley, 1994.

[78] W. Richard Stevens. RFC 2001: TCP slow start, congestion avoidance, fast retransmit and fast recovery algorithms, Jan 1997.

[79] W. Richard Stevens. *UNIX network programming*. Prentice Hall, second edition, 1998.

[80] Surveyor Home Page. `http://www.advanced.org/surveyor/`.

[81] Cisco Systems. Cisco IOS Quality of Service Solutions. `http://www.cisco.com/warp/public/cc/cisco/mkt/ios/tech/tch/qosio_wp.htm`.

[82] Cisco Systems. NetFlow Switching Overview. `http://www.cisco.com/warp/public/732/netflow/`.

[83] Gene Trent and Mark Sake. WebStone: The First Generation in HTTP Server Benchmarking. `http://www.sgi.com/Products/WebFORCE/WebStone/paper.html`.

[84] Henk Uijterwaal. Comparing Two Implementations of the Delay and Loss Metrics (ANS and RIPE-NCC). `http://www.ripe.net/test-traffic/Talks/9903_ietf/index.html`.

[85] Henk Uijterwaal. Test Traffic Project Homepage. `http://www.ripe.net/test-traffic/`.

[86] Universität zu Köln. Bericht für das Jahr 1998, Abteilung Kommunikationsnetze. In *Kooperative Datenverarbeitung an der Universität zu Köln*. Der Rektor der Universität zu Köln, 1999.

[87] Larry Wall, Tom Christiansen, and Randal L. Schwartz. *Programming Perl*. O'Reilly & Associates, second edition, 1996.

[88] Gary R. Wright and Richard W. Stevens. *TCP/IP Illustrated*, volume 2: the implementation. Addison-Wesley, 1995.

[89] Werner Zorn and Egbert Fridrich. Erste Näherung: Leistungsmessungen im Internet – von deutschen Service Providern aus. *iX*, pages 50–65, 1995. Verlag Heinz Heise GmbH & Co KG.

Erklärung

Ich versichere, dass ich die von mir vorgelegte Dissertation selbständig angefertigt, die benutzten Quellen und Hilfsmittel vollständig angegeben und die Stellen der Arbeit — einschließlich Tabellen, Karten und Abbildungen —, die anderen Werken im Wortlaut oder dem Sinn nach entnommen sind, in jedem Einzelfall als Entlehnung kenntlich gemacht habe; dass diese Dissertation noch keiner anderen Fakultät oder Universität zur Prüfung vorgelegen hat; dass sie — abgesehen von unten angegebenen Teilpublikationen — noch nicht veröffentlicht worden ist sowie, dass ich eine solche Veröffentlichung vor Abschluss des Promotionsverfahrens nicht vornehmen werde. Die Bestimmungen dieser Promotionsordnung sind mir bekannt. Die von mir vorgelegte Dissertation ist von Professor Dr. R. Schrader betreut worden.

Teilpublikationen

- Martin Horneffer. Experience with Tests of Connectivity. Vortrag auf der 39. IETF-Tagung am 11.8.1997, München. Siehe auch `http://www.advanced.org/IPPM/archive.2/0333.html`.
- Martin Horneffer. Methods for performance-analysis of internet access points. In *Proceedings of the TERENA Networking Conference '98*, volume 30 of *Computer Networks and ISDN Systems*, pages 1607–1615. Elsevier Science, September 1998.
- Martin Horneffer. Assessing internet performance metrics using large-scale TCP-syn based measurements. In *The First Passive and Active Measurement Workshop (PAM 2000)*, pages 51–59. Department of Computer Science, University of Waikato, New Zealand, April 2000. ISBN 0-909007-20-9, `http://pam2000.cs.waikato.ac.nz/final_program.htm`.

Lebenslauf

Persönliche Daten

Name	Ernst Martin Horneffer
Geburtsort	Berlin – Schöneberg
Geburtstag	17. April 1968
Familienstand	ledig
Staatsangehörigkeit	deutsch

Schulbildung/Studium

1974 – 1979	Grundschule „Stenzelbergschule“ in Königswinter
1979 – 1988	„Gymnasium am Oelberg“ in Königswinter, Abschluss Abitur
1988 – 1989	Grundwehrdienst
10/1989 – 8/1995	Studium der Elektrotechnik, Fachrichtung „Technische Informatik“ an der Rheinisch-Westfälischen Technischen Hochschule Aachen, Abschluss Diplom-Ingenieur der Elektrotechnik

Praktika

7/1988 – 9/1988	Industrie-Grundpraktikum im Heizkraftwerk Karlstraße, Stadtwerke Bonn
5/1994 – 7/1994	Industrie-Fachpraktikum im Televerkets Forskningsinstitutt, jetzt „Telenor Research“, Kjeller, Norwegen

Berufstätigkeit

2/1992 – 8/1995	Studentische Hilfskraft am Lehrstuhl für Kommunikationsnetze der RWTH Aachen; Design und Entwicklung objekt-orientierter Software zur Visualisierung ereignisorientierter Simulationssysteme
seit 9/1995	Wissenschaftlicher Mitarbeiter am Regionalen Rechenzentrum der Universität zu Köln, Abteilung Kommunikationsnetze