

Studiengang: Informatik
Prüfer : Prof. Dr. Kurt Rothermel
Betreuer : Dipl. Inf. Daniel Herrscher

Begonnen am : 15. Mai 2002
Beendet am : 14. November 2002

CR-Klassifikation : C.4, G.3, I.6.5

Studienarbeit-Nr. 1856

Realistische Lastspeisung in Emulationsszenarien

Ulrich Wurst

Institut für Parallele und Verteilte Systeme
Universität Stuttgart
Breitwiesenstr. 20-22
D-70565 Stuttgart

Kurzfassung

Diese Arbeit behandelt die synthetische Lasterzeugung und -einspeisung in Rechnernetze, insbesondere in Hinblick auf Emulationsszenarien. Basierend auf Modellen zur Beschreibung von Netzlast aus anderen Arbeiten wird ein Java-Rahmenwerk zur Erzeugung von Last entwickelt, in das mehrere anwendungsspezifische Lastmodule integriert werden. Diese erzeugen Netzlast mit denselben Eigenschaften, wie sie in verschiedenen Internetanwendungen vorgefunden werden.

Inhaltsverzeichnis

1	Einleitung.....	5
1.1	Motivation	5
1.2	Umfang der Arbeit	5
2	Grundlagen.....	7
2.1	Das TCP/IP Schichtenmodell	7
2.2	Das Network Emulation Testbed	8
2.3	Statistik	10
2.4	Klassifikation von Anwendungen	14
3	Modellbildung	15
3.1	Wiedergabe aufgezeichneter Daten	15
3.2	Generierung eines aggregierten Datenstromes	16
3.3	Generierung von Last nach dem white box modell	17
3.4	Generierung von Last nach statistischen Modellen von Anwendungsprotokollen	18
3.5	Gegenüberstellung empirischer und analytischer Modelle	19
3.6	Ansätze zur Modellgewinnung	20
4	Modelle verschiedener Anwendungen	22
4.1	Verwandte Arbeiten	22
4.2	FTP	23
4.3	HTTP	27
4.4	Digitales Video	30
4.5	TELNET	34
4.6	Onlinespiele	36
5	Architektur der Software.....	39
5.1	Benutzungsoberfläche (GUI)	39
5.2	Knotensteuerung	40
5.3	Lasterzeugungs-Module	40
6	Implementierung der Statistikfunktionen	41
6.1	Inverse Transformation (exponential-, Pareto- und Gumbel-Verteilung)	41
6.2	Annäherung einer geschlossenen Funktion (Normal-Verteilung)	42
6.3	Box-Müller Verfahren und Polarverfahren	42
6.4	Generierung von lognormalverteilten Ereignissen	43
6.5	acceptance-rejection Methode (Poisson-Verteilung)	43
6.6	Generierung von zipfverteilten Zufallszahlen	44
6.7	Inverse Transformation bei heuristischen Verteilungen	44
7	Diskussion der Anwendungsmodelle.....	46
7.1	FTP	46
7.2	HTTP	46
7.3	Telnet	47
7.4	MPEG Videostrom	47
7.5	Onlinespiele	48
8	Zusammenfassung und Ausblick.....	49
9	Literaturangaben	50

Abbildungsverzeichnis

Abb. 1	Die TCP/IP und ISO/OSI Modelle im Vergleich	7
Abb. 2	Das Network Emulation Testbed [Herrscher et al. 2002]	9
Abb. 3	Protokollstapel eines NET-Knotens [Herrscher, Rothermel 2002]	9
Abb. 4	Dichte- und Verteilungsfunktion der Standardnormalverteilung	11
Abb. 5	Ebenen der Kommunikation [Charzinski et al. 2002]	15
Abb. 6	GenSyn-Modell eines HTTP-Clients [Heegaard 2000]	17
Abb. 7	Modell für Telnet [Danzig et al 1992]	18
Abb. 8	Ablauf einer FTP-Verbindung	23
Abb. 9	Entscheidung der Burstzugehörigkeit	26
Abb. 10	Wahrscheinlichkeitsverteilungen des FTP Modells	26
Abb. 11	Ablauf einer HTTP-Sitzung	29
Abb. 12	Abhängigkeiten zwischen I-, P- und B-Frames eines MPEG-Stroms	30
Abb. 13	Markov Modell für dynamische GOP Sequenzen [Agrawal et al. 1997].	31
Abb. 14	Charakteristische Größen in Videos [Krunz et al. 1996]	32
Abb. 15	Ablauf einer Telnet-Verbindung	35
Abb. 16	Länge von Kommandos in Telnetsitzungen	36
Abb. 17	Ablauf der Onlinespiele-Kommunikation	37
Abb. 18	Statistische Verteilungen in Counterstrike [Faerber 2002]	38
Abb. 19	Das MVC-Konzept	39
Abb. 20	Architektur eines Lasterzeugungs-Moduls	40
Abb. 21	Methode der Inversen Transformation	41
Abb. 22	Inverse Transformation bei heuristischen Modellen	45

1 Einleitung

Diese Arbeit behandelt die Erzeugung und Einspeisung von Netzlast in Emulationsszenarien. Die künstlich erzeugte Netzlast soll dabei realistische Eigenschaften besitzen. Dies bedeutet, dass sie dasselbe statistische Verhalten wie die nachgebildeten Anwendungen aufweisen muss.

1.1 Motivation

Zur Entwicklung neuer Netzwerksoftware wie verteilter Anwendungen kann es hilfreich sein, deren Verhalten in bezug auf besondere Netzwerkszenarien wie eine verringerte Bandbreite, lange Paketlaufzeiten oder eine erhöhte Paketfehlerrate zu untersuchen. Zur Nachbildung bestimmter Netzwerkszenarien, in denen neue Anwendungen untersucht werden können, stehen zwei Möglichkeiten bereit.

Mit Hilfe von Simulationssoftware kann das Verhalten von Kommunikationssystemen mittels mathematischer Modelle nachgebildet werden. Durch Veränderung einzelner Parameter der Modelle kann so eine längere Paketlaufzeit simuliert werden, oder es kann die Bandbreite festgelegt werden. Nachteil dieser Methode ist, dass die zu testende Software ebenfalls in ein mathematisches Modell umgewandelt werden muss. Dies bringt einen hohen Zusatzaufwand mit sich, insbesondere da das Modell gegebenenfalls mit jeder Weiterentwicklung der Software angepasst werden muss. Zudem besteht in der Entwicklung des Modells die Gefahr zusätzlicher Fehlerquellen, wenn das Verhalten der Software nicht korrekt abgebildet wird.

Um diese Probleme zu umgehen kann anstatt der Netzsimulation die Netzemulation verwendet werden. Bei der Netzemulation wird das Kommunikationssystem, in dem der Test durchgeführt werden soll, nahezu unverändert übernommen. Es wird lediglich eine Emulationsschicht zwischen das Treibermodul der Netzwerkkarte und der Netzwerkschicht (im Allgemeinen IP) eingeführt. Diese Emulationsschicht erlaubt das Nachbilden der gewünschten Netzwerkparameter wie Verzögerung, Paketlaufzeit oder Paketverlustrate. Durch die Verwendung des ansonsten unveränderten Kommunikationssystems bei der Netzemulation wird die oben genannte Beschränkung der Netzsimulation umgangen. Insbesondere kann die zu testende Software ohne Anpassungen übernommen und ihr Verhalten bezüglich spezieller Netzwerksituationen überprüft werden.

Eine isolierte Betrachtung des zu testenden Programms im Kommunikationssystem ist jedoch in der Regel nicht realistisch, da in realen Netzen immer mehrere Anwendungen gleichzeitig kommunizieren. Daher kann es zum realistischen Test von Software notwendig sein, in das Emulationsnetzwerk, in dem der Test stattfindet, Hintergrundlast einzuspeisen, die die Netzlast der anderen im Netz aktiven Anwendungen simuliert.

1.2 Umfang der Arbeit

Diese Arbeit evaluiert verschiedene Methoden der Lasterzeugung in Hinblick auf deren Eignung für Emulationsszenarien und in Hinblick auf eine realitätsnahe Nachbildung der zu simulierenden Anwendungen. Weiterhin wird ein eigener Lastgenerator entwickelt, der auf Grundlage von Lastmodellen verschiedener Internet-Anwendungen, die anderen Arbeiten entnommen wurden, deren Netzlast nachbildet.

Die Lastmodelle enthalten statistische Modelle der charakteristischen Eigenschaften der jeweiligen Anwendungen. Dazu können Paketgröße, Verbindungsdauer oder Größe der

übertragenen Dateien zählen, die üblicherweise mit Hilfe von Messungen in realen Netzen entwickelt wurden. Zusätzlich enthalten die Lastmodelle Aussagen über die Zusammenhänge der charakteristischen Eigenschaften der jeweiligen Anwendung.

Eine Lasterzeugung mit realistischem Verhalten bezüglich realer Anwendungen ist dann möglich, indem Datenpakete zwischen Rechnern versandt werden, deren charakteristische Eigenschaften wie Paketgröße oder Paketankunftsrate statistisch mit den gemessenen statistischen Werteverteilungen der nachzubildenden Anwendungen übereinstimmen.

2 Grundlagen

Dieses Kapitel behandelt Grundlagen aus den Bereichen Rechnernetze und Statistik, die im weiteren Verlauf dieser Arbeit benötigt werden.

2.1 Das TCP/IP Schichtenmodell

Um die Komplexität von Kommunikationssystemen zu reduzieren, sind diese üblicherweise in mehrere Schichten unterteilt [Tanenbaum 1996]. Jede Schicht greift über standardisierte Schnittstellen (SAP = Service Access Point) auf die jeweils Darunterliegende zu. Dies vereinfacht die Implementierung von Kommunikationssystemen, da man sich auf einzelne Komponenten des Systems beschränken kann, und die Funktionen der Schichten ober- und unterhalb der zu Implementierenden nicht verändert werden müssen. Üblicherweise können in jeder Schicht unterschiedliche Protokolle verwendet werden. Eine Auswahl mit je einem Protokoll jeder verwendeten Schicht wird als Protokollstapel bezeichnet.

Das derzeit am häufigsten verwendete Schichtenmodell ist das TCP/IP Referenzmodell. Es weist im Vergleich zum, vor allem bei theoretischen Betrachtungen verwendeten, ISO/OSI Modell mehrere Vereinfachungen auf, da vor Allem aus Gründen der höheren Geschwindigkeit beim TCP/IP Modell einige Aspekte der Kommunikation nicht implementiert wurden.

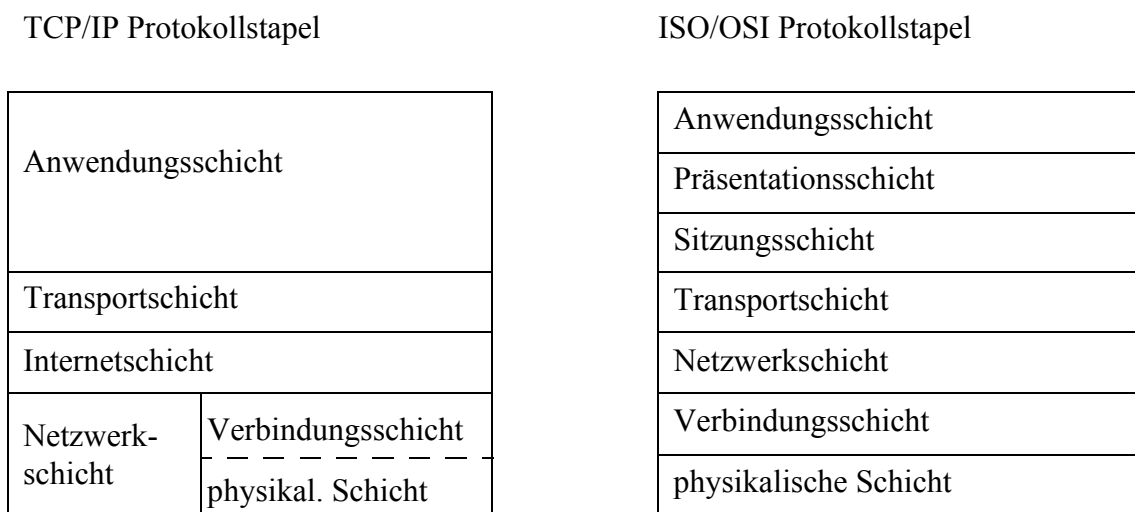


Abb. 1 Die TCP/IP und ISO/OSI Modelle im Vergleich

Das TCP/IP Modell betrachtet die Kommunikation zwischen zwei Systemen auf fünf Schichten. Die beiden untersten Schichten des Protokollstapels, die in der Netzwerkschicht zusammengefasst sind, werden im TCP/IP Modell nicht explizit beschrieben, lediglich die SAPs zur Verbindung zwischen Internet- und Netzwerkschicht sind spezifiziert [Tanenbaum 1996].

Intern ist die Netzwerkschicht in die physikalische Schicht und die Verbindungsschicht zweigeteilt. Die Aufgabe der physikalischen Schicht liegt in der Übermittlung eines Bitstroms zwischen zwei direkt verbundenen Kommunikationspartnern. Die physikalische Schicht spezifiziert dazu physikalische und funktionale Parameter zur Übertragung und Erkennung einer logischen „1“ und einer logischen „0“ über ein Trägermedium (Kupferkabel, Radiowellen usw.). Die darüberliegende Verbindungsschicht hat die Aufgabe, eine zuverlässige Übertragung von Datenrahmen (Frames) über den zunächst unzuverlässigen

Bitstrom zu gewährleisten. Dazu werden einerseits Methoden zum gemeinsamen Zugriff auf von mehreren Stationen verwendete Medien (shared media) definiert, die beispielsweise zum koordinierten Zugriff auf einen Funkkanal erforderlich sind. Weiterhin spezifiziert die Verbindungsschicht Synchronisationsmaßnahmen zur Erkennung von Rahmenanfang und -ende, sowie Algorithmen zur Fehlererkennung und -korrektur. Beispiele für die Protokolle der Netzwerkschicht sind Ethernet (das beide Schichten abdeckt) oder PPP über HDLC, wobei ersteres ein Protokoll der Verbindungsschicht und letzteres das physikalische Protokoll darstellt.

Während die Netzwerkschicht die Kommunikation zwischen zwei benachbarten Stationen ermöglicht, besteht die Aufgabe der Internetschicht in der Vermittlung zwischen mehreren Netzen. Die Hauptaufgabe der Internetschicht besteht in der Wegewahl basierend auf Adressinformationen der Datenpakete. Die Internetschicht arbeitet verbindungslos, somit können zwei Datenpakete zwischen zwei Stationen über verschiedene Wege übermittelt werden und können in beliebiger Reihenfolge am Ziel ankommen. Auch die fehlerfreie Übermittlung selbst wird von der Internetschicht nicht garantiert. Im TCP/IP Modell wird auf dieser Schicht das IP-Protokoll verwendet.

Die über der Internetschicht liegende Transportschicht dient zum Aufbau von Ende-zu-Ende-Verbindungen zwischen zwei Kommunikationspartnern. Transportkanäle können unterschiedliche Eigenschaften besitzen. Das am häufigsten verwendete Protokoll der Transportschicht ist das verbindungsorientierte TCP-Protokoll, das einen zuverlässigen Transportkanal zur Übertragung eines Bytestromes zwischen zwei Kommunikationspartnern ermöglicht. Ebenfalls gebräuchlich ist das UDP-Protokoll, das einen verbindungslosen Transportkanal zwischen zwei Kommunikationspartnern zur Verfügung stellt. Weiterhin sind auch Punkt-zu-Mehrpunkt-Verbindungen oder verschlüsselte Verbindungen mögliche Transportkanäle dieser Schicht. Ein Teilnehmer eines Kommunikationsnetzes kann mehrere gleichzeitige Transportkanäle aufbauen.

Die oberste Schicht des TCP/IP Modells wird Anwendungsschicht genannt. Dabei handelt es sich jedoch nicht um die Anwendung selbst, sondern um ein Protokoll zum Zugriff auf Funktionen der Anwendung. Ein Beispiel für ein Protokoll dieser Schicht ist das SMTP-Protokoll zur Übermittlung von Emails. Es ist nicht von Bedeutung, ob Sendmail, Lotus Notes oder MS-Exchangeserver als Mailserver verwendet wird, da mit allen Mailservern über das SMTP-Protokoll kommuniziert werden kann. Weitere Beispiele für Protokolle dieser Schicht sind FTP, TELNET oder HTTP.

Die Sitzungsschicht und Präsentationsschicht, die im ISO/OSI Modell spezifiziert sind, sind im TCP/IP Modell nicht vorhanden.

2.2 Das Network Emulation Testbed

Das Network Emulation Testbed (NET) des Instituts für Parallele und Verteilte Systeme der Universität Stuttgart ist ein flexibel vernetzter PC-Cluster zur Nachbildung beliebiger Netztopologien und deren Eigenschaften. Die Knoten des NET sind mittels eines VLAN-fähigen Switch verbunden, so dass beliebige Gruppen von Knoten gebildet werden können. VLANs sind eine Möglichkeit zur Virtualisierung von Rechnernetzen auf der Schicht 2. Dazu werden den Netzwerkschnittstellen der Netzwerkknoten sogenannte VLAN-Tags zugewiesen. Alle Schnittstellen mit demselben VLAN-Tag befinden sich in einem virtuellen lokalen Netzwerk. Zwei Schnittstellen mit unterschiedlichem VLAN-Tag können keine direkte Kommunikation zueinander aufbauen, selbst wenn sie mit demselben Switch verbunden sind, da aufgrund der unterschiedlichen VLANs keine direkte Schicht 2 Verbindung besteht. Sie müssen daher über Schicht 3 verbunden werden, also über einen (oder mehrere) Router.

Somit lassen sich mit Hilfe von VLANs mehrere voneinander getrennte Schicht-2-Netze aufbauen, obwohl alle beteiligten Netzwerkkomponenten mit einem einzigen Switch verbunden sind. Dies ist Voraussetzung für die softwaregesteuerte Bildung von Knotengruppen um Netzwerktopologien definieren zu können, ohne eine Änderung an der physikalische Verkabelung vorauszusetzen.

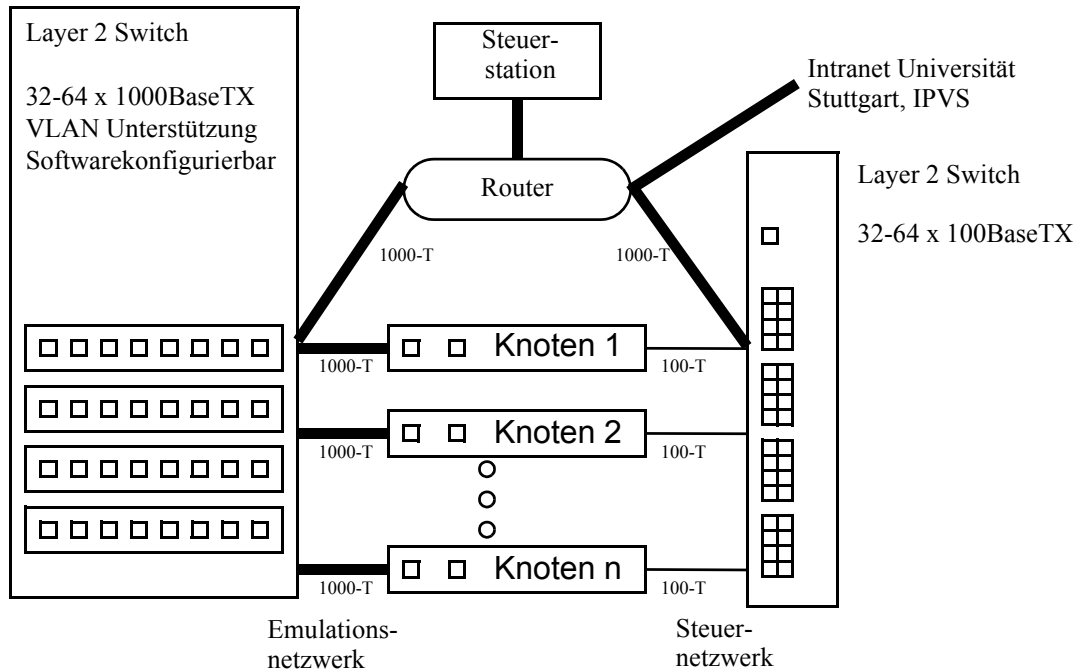


Abb. 2 Das Network Emulation Testbed [Herrscher et al. 2002]

Neben der Möglichkeit, mit Hilfe von VLANs beliebige Gruppen von Knoten zusammenzufassen, können im Network Emulation Testbed mittels Emulationswerkzeugen Parameter wie Ausbreitungsverzögerung, Bandbreite und Paketverlustrate variiert werden. Dazu wird zwischen dem TCP/IP Protokoll und dem Hardwaretreiber der Netzwerkkarte eine Emulationsschicht eingeführt, die die nachzubildenden Parameter wie Paketlaufzeit oder Bandbreite beeinflusst. In Zukunft sollen auch Möglichkeiten zur Implementierung von Netzen mit gemeinsamem Medium und mobilen Netzen integriert werden.

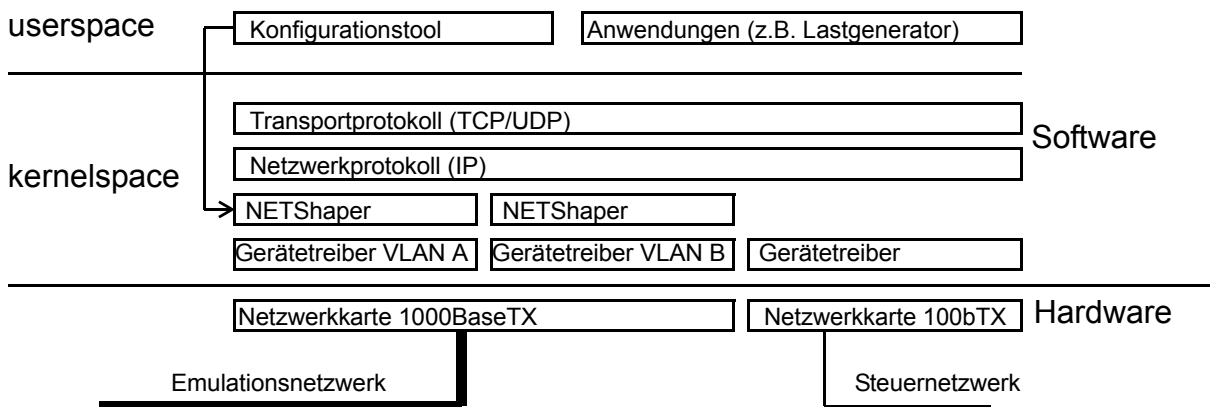


Abb. 3 Protokollstapel eines NET-Knotens [Herrscher, Rothermel 2002]

Im Vergleich zur reinen Software-Netzsimulation, wie beispielsweise dem Netzesimulator ns-2, bietet die Emulation von Netzen den Vorteil von realistischeren Ergebnissen. Während bei Simulatoren alle Eigenschaften des Kommunikationssystems durch mathematische Modelle nachgebildet werden müssen, werden bei der Netzwerk-Emulation große Teile des Kommunikationssystems, wie beispielsweise der TCP/IP-Stack, die CPU und die Anwendungssoftware in ihren unveränderten Versionen verwendet. Da bei größeren Szenarien die Rechenzeit in Simulationen stark anwächst, müssen dort oftmals vereinfachte Modelle verwendet werden, was in einem weniger realistischen Ergebnis resultiert. Ein weiterer Vorteil der Netzemulation im Vergleich zu deren Simulation liegt im geringeren Aufwand im Hinblick auf die Anpassung der nachzubildenden Kommunikationssysteme. Während zur Netzsimulation die nachzubildenden Kommunikationssysteme in ein mathematisches Modell überführt werden müssen, können sie bei der Netzemulation direkt übernommen werden. Lediglich die Emulationsschicht muss für jedes beteiligte System zur Verfügung stehen.

Ein mögliches Problem von Netzemulationssystemen sind die hohen Kosten, da je nach maximaler Größe der nachzubildenden Kommunikationssysteme viele Knoten, die in diesem Fall physikalische Rechner sind, benötigt werden. Ein Simulationsszenario dagegen kann ein komplexes Kommunikationssystem auch auf einem einzigen Rechner abbilden. Dieser muss jedoch je nach Umfang der Simulation sehr leistungsfähig sein.

2.3 Statistik

Die Nachbildung von Anwendungen zur Lasterzeugung erfordert eine Generierung von Datenpaketen mit denselben statistischen Eigenschaften, wie sie bei den realen Anwendungen beobachtet werden.

Die Statistik ist eine mathematische Disziplin, die versucht aus den beobachteten Daten zufälliger Prozesse oder Ereignisse Schlüsse über deren Wahrscheinlichkeiten zu ziehen. Die Wahrscheinlichkeitstheorie liefert dazu die mathematischen Grundlagen. Beide Teilbereiche zusammen ergeben den Bereich der Stochastik und bieten Methoden zur Beschreibung von zufälligen Ereignissen.

2.3.1 Zufallsvariable

Eine Zufallsvariable ist eine Größe, die bei jeder Messung einen Wert aus einem vorgegebenen Wertebereich annimmt. Die einzelnen Werte des Wertebereiches können dabei mit unterschiedlichen Wahrscheinlichkeiten vorkommen. Beispiele für Zufallsvariablen sind die übertragene Datenmenge einer FTP-Verbindung oder die Dauer einer Telnet-Sitzung. Eine Instanz einer Zufallsvariablen ist das Ergebnis einer einzelnen Messung. Eine Messung wird auch als Versuch bezeichnet.

Eine Zufallsvariable wird mit X , die i -te Instanz einer Versuchsreihe der Zufallsvariablen X mit x_i bezeichnet. Falls nicht weiter ausgeführt, soll eine Versuchsreihe aus n Versuchen bestehen.

2.3.2 Modell

Als Modell einer Zufallsvariablen wird die angenommene Verteilung einer Zufallsvariablen bezeichnet, also ihren Wertebereich und die jeweilige Wahrscheinlichkeit, dass die Zufallsvariable einen bestimmten Wert annimmt. Es werden unabhängige und korrelierte Modelle unterschieden. Bei unabhängigen Modellen sind die Instanzen einer Zufallsvariablen unab-

hängig voneinander und besitzen damit eine identische Verteilung. Somit ist die Wahrscheinlichkeit, dass die Instanz einer Zufallsvariable einen bestimmten Wert annimmt, unabhängig von den Ergebnissen vorhergegangener Versuche.

In korrelierten Modellen ist dagegen die Wahrscheinlichkeitsverteilung einer Instanz der Zufallsvariablen von den anderen Instanzen der Zufallsvariablen abhängig. Die Autokorrelationsfunktion gibt die Beziehung zwischen den Instanzen der Zufallsvariablen an.

2.3.3 Dichte und Verteilung

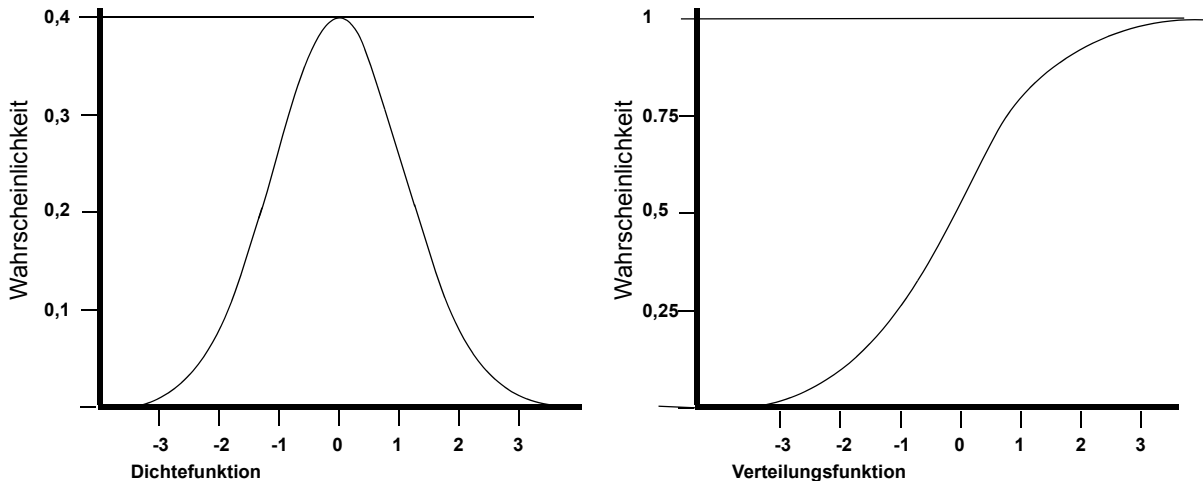


Abb. 4 Dichte- und Verteilungsfunktion der Standardnormalverteilung

Die Dichtefunktion $f(x)$ eines Modells ist definiert als

$$f(x) = P(X = x) ,$$

also als die Wahrscheinlichkeit, dass ein gewähltes Ereignis $X=x$ eintritt. Die (kumulierte) Verteilungsfunktion F (engl. CDF=Cumulative Density Function) einer Zufallsvariable ist definiert als

$$F(x) = P(X \leq x) .$$

Damit gibt $F(x)$ die Wahrscheinlichkeit an, dass eine Instanz der Zufallsvariable einen Wert kleiner oder gleich x annimmt. Zwei daraus resultierende Eigenschaften der Verteilungsfunktion sind, dass sie den Wertebereich $[0..1]$ besitzt und die Funktion für wachsende x monoton gegen 1 strebt. In der weiteren Arbeit werden folgende Verteilungen verwendet:

Verteilung	Wahrscheinlichkeits-/ Dichtefunktion	Verteilungsfunktion
Normal	$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$	$F(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \int_{-\infty}^x e^{-\frac{1}{2}\left(\frac{\xi-\mu}{\sigma}\right)^2} d\xi$

Verteilung	Wahrscheinlichkeits-/ Dichtefunktion	Verteilungsfunktion
Extremal (Gumbel)	$f(x) = \frac{1}{b} e^{-\frac{x-x_0}{b}} e^{-\exp\left(-\frac{x-x_0}{b}\right)}$	$F(x) = e^{-\exp\left(-\frac{x-x_0}{b}\right)}$
Pareto	$f(x) = \frac{\alpha}{\beta} \left(\frac{\beta}{x}\right)^{\alpha+1}$	$F(x) = 1 - \left(\frac{\beta}{x}\right)^\alpha$
Poisson	$f(x) = e^{-\lambda} \frac{\lambda^x}{x!}$	$F(x) = \begin{cases} e^{-\lambda} \sum_{i=0}^x \frac{\lambda^i}{i!}, & x \geq 0 \\ 0, & x < 0 \end{cases}$
Exponential	$f(x) = \lambda e^{-\lambda x}$	$F(x) = \begin{cases} 0, & x < 0 \\ 1 - e^{-\lambda x}, & x \geq 0 \end{cases}$
Uniform (Gleichverteilung)	$f(x) = \begin{cases} \frac{1}{b-a}, & a \leq x \leq b \\ 0, & \text{sonst} \end{cases}$	$F(x) = \begin{cases} 0, & x < a \\ \frac{x-a}{b-a}, & a \leq x \leq b \\ 1, & x > b \end{cases}$
Geometrisch	$f(x) = p(1-p)^{x-1}$	$F(x) = 1 - (1-p)^x$

Zusätzlich zu diesen Dichtefunktionen werden die Lognormal- und die Loggumbel-Funktion verwendet. Die Log-Funktion der Normal-Verteilung ist definiert als:

$$Y = \log X \sim \text{Normal} \Rightarrow X \sim \text{Lognormal}$$

Falls $\log(X)$ also Normalverteilt ist, so ist X selbst Lognormalverteilt. Bei der Erzeugung von Lognormalverteilten Zufallszahlen macht man sich dies zu Nutze, indem man eine normalverteilte Zufallszahl berechnet, aus der man mittels der Umkehrfunktion $\log^{-1}(x) = e^x$ eine Lognormalverteilte Zufallszahl erzeugt. Neben der Lognormal-Dichtefunktion zur Basis e wird auch die Log2normal-Funktion zur Basis 2 verwendet.

2.3.4 Verteilungsparameter

Alle zuvor genannten Verteilungen werden durch eine oder mehrere Konstanten charakterisiert. Die Parameter der Normal-Verteilung und Lognormal-Verteilung sind der Mittelwert und die Standardabweichung.

Der Mittelwert einer Menge von Werten x_i berechnet sich als:

$$\hat{\bar{x}} = \sum_{i=1}^n x_i / n$$

Die Standardabweichung als:

$$\hat{\sigma}_x = \sqrt{\frac{\sum_{i=1}^n (x_i - \hat{\bar{x}})^2}{(n-1)}}$$

Eine exponentielle Verteilung wird durch die Rate λ , eine Extremal-Verteilung durch Formparameter b und Ortsparameter x_0 und eine Pareto-Verteilung ebenfalls durch Formparameter α und Ortsparameter β charakterisiert, wobei β der kleinste Wert der Verteilung ist. Der Parameter λ der Poisson-Verteilung wird ebenfalls als Formparameter bezeichnet. Die Verteilungsparameter für die Beschreibung von Verkehrslasten werden üblicherweise experimentell aus Messungen in Netzwerken bestimmt.

2.3.5 Quantile

Die Quantile x_p einer Verteilung sind Rangmaßzahlen, die den Wert aus dem Wertebereich einer Verteilung angeben, für den der Anteil p der Beobachtungen kleiner als die Quantile ist. Beispielsweise gibt die Quantile $x_{0.5}$ den Wert des Wertebereichs an, für den die Hälfte der Beobachtungen kleiner als dieser Wert ist, also den Median. Die Quantile $x_{0.25}$ gibt den Wert an, für den ein Viertel der beobachteten Werte kleiner als dieser Wert sind. Quantilen sind für die Erzeugung von Zufallszahlen, die einer vorgegebenen Verteilung entsprechen, von Bedeutung (Kapitel 6).

2.3.6 „heavy tailed“ Verteilungen

Als „heavy tailed“ werden Verteilungen bezeichnet, die der Gesetzmäßigkeit

$$P[X > x] \sim x^{-\alpha} \quad 0 < \alpha < 2$$

genügen. Dabei wird α als Formparameter bezeichnet. Verteilungen, die „heavy tailed“ sind, sind stark von außergewöhnlich großen Werten abhängig. Beispielsweise verursachen in [Paxson 1994] weniger als 1% der FTP-Dateiübertragungen über 50% des Datenverkehrs.

Die Pareto-Verteilung ist die in dieser Arbeit verwendete „heavy tailed“ Verteilung. Der Formparameter ist auf den Bereich zwischen 1 und 2 festgelegt, der zusätzliche Parameter β der Pareto-Verteilung wird als Ortsparameter bezeichnet und gibt den kleinsten Wert der Dichtefunktion an.

2.3.7 Zusammengesetzte Verteilungen

Oftmals ist die Wiedergabe einer gemessenen Verteilung nicht mit einer einzigen Verteilungsfunktion möglich. In diesem Fall kann es sinnvoll sein, zwei oder mehrere Verteilungen zu verwenden, und diese zur einer mehrteiligen Verteilungsfunktion zusammensetzen. Ein Beispiel für eine zusammengesetzte Verteilung ist die Intra-Burst Zeit zwischen zwei Dateiübertragungen im FTP Protokoll. Der Zeitraum zwischen 0 und 2,5 Sekunden wird mit einer exponential-Verteilung nachgebildet, der Zeitraum zwischen 2,5 und 4 Sekunden mit einer uniformen Verteilung.

Üblicherweise wird die Trennung zwischen den Verteilungen jedoch nicht mit Abszissenwert angegeben, sondern mit dem Ordinatenwert im Diagramm der Verteilungsfunktion, an dem die Trennung durchgeführt wird. Dies ist zweckmäßig, da bei allen Verteilungsfunktionen die Ordinate zwischen 0 und 1 (bzw. zwischen 0 und 100%) liegt, und somit einfacher dargestellt werden kann, als ein Abszissenwert, der im Allgemeinen keine Größenbeschränkungen hat.

2.4 Klassifikation von Anwendungen

Die unterschiedlichen im Internet vertretenen Anwendungen bzw. deren Protokolle besitzen charakteristische Eigenschaften, die durch einen Lastgenerator nachgebildet werden müssen. Bei der Betrachtung der einzelnen Anwendungen sind die Klassen Dialog- und Massendaten zu unterscheiden.

2.4.1 Dialog-Kommunikation (auch: interaktive Verbindungen)

In die Klasse der Dialog-Kommunikation fallen beispielsweise die Anwendungen TELNET, SSH und X11. Diese Anwendungen sind dadurch gekennzeichnet, dass die Übertragungsparameter wie Verbindungsdauer und Paketankunftsrate weit weniger durch das Übertragungssystem als vielmehr durch den Anwender bestimmt werden. Da insbesondere TELNET sehr kleine Datenpakete verschickt, haben Veränderungen der verfügbare Bandbreite nur geringe Auswirkung auf den Datenverkehr. Die Parameter Paketgröße und Ankunftsrate sind daher weitestgehend vom Netzwerk unabhängig.

2.4.2 Massendaten-Kommunikation (Bulk-Transfer)

Im Gegensatz zu der Dialog-Kommunikation sind bei der Übermittlung von Massendaten, wie im HTTP oder dem FTP Protokoll, die Parameter des darunterliegenden Netzwerkes entscheidend für die Parameter der Übertragung. Insbesondere die Dauer der Verbindung und die Paketankunftsrate sind direkt abhängig von der Bandbreite der Verbindung. Weiterhin ist die Paketgröße abhängig vom Netzwerk, da vor allem Pakete mit der maximal zulässigen Größe (MTU = Maximum Transfer Unit) übermittelt werden. Lediglich die Menge der zu übertragenden Daten ist direkt vom Anwender abhängig.

3 Modellbildung

Modelle zur Lasterzeugung können - je nach Einsatzgebiet - nach verschiedenen Grundsätzen gebildet werden. [Heegaard 2000] nennt folgende Möglichkeiten:

1. Trace: die Aufzeichnung und Wiedergabe von vollständigen Datenströmen beziehungsweise die Aufzeichnung der Paketheader und die Rekonstruktion eines ähnlichen Datenstromes
2. Aggregate generator ("blackbox"): die Generierung eines IP-Datenstromes, entsprechend eines stochastischen Modells auf Netzwerk-Ebene (IP)
3. User behaviour model ("white box"): die Generierung von Last durch die jeweiligen Anwendungsprogramme, wobei lediglich das Verhalten des Anwenders simuliert wird. Die Kommunikation selbst findet mittels weitestgehend unveränderter Anwendungssoftware statt.

Eine weitere Möglichkeit, die Heegaard nicht nennt, und die Gegenstand dieser Arbeit sein wird, besteht in der Modellierung eines Datenstromes entsprechend eines stochastischen Modells auf Anwendungsebene. Im Gegensatz zum blackbox-Generator wird hier nicht ein aggregierter Datenstrom erzeugt, der alle verwendeten Netzwerkprotokolle beinhaltet, und im Gegensatz zum whitebox-Modell wird nicht auf vollständige Anwendungsprogramme zurückgegriffen.

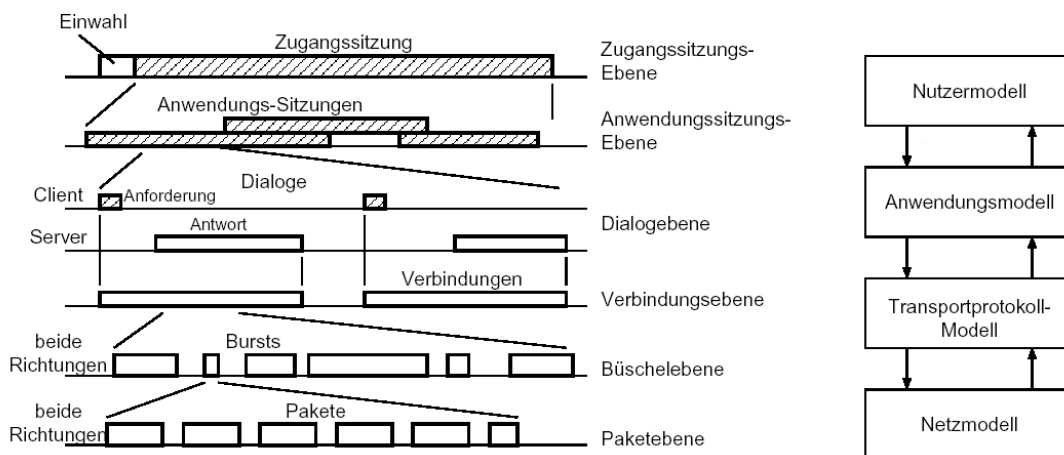


Abb. 5 Ebenen der Kommunikation [Charzinski et al. 2002]

Last kann auf unterschiedlichen Ebenen der Kommunikation betrachtet werden. Ein Tracemodell betrachtet beispielsweise lediglich die Paketebene und enthält keine Informationen über darüberliegende Schichten. Dagegen modelliert ein white-box Modell lediglich die Anwendungs-Sitzungen und verwendet für die Generierung von Ereignissen (Verbindungen, Pakete usw.) auf niedrigeren Schichten modifizierte Anwendungsprogramme.

3.1 Wiedergabe aufgezeichneter Daten

Die Verwendung von zuvor aufgezeichneten Datenströmen kann verwendet werden, um reproduzierbare Netzlasten zu erzeugen. Alternativ zur vollständigen Aufzeichnung und Wiedergabe der Datenströme können, beispielsweise mit Hilfe von tcpdump [tcpdump], aus dem Datenstrom des Netzwerkes die Paket-Header extrahiert und abgespeichert werden.

Aus den Headern kann zur Lasterzeugung ein Datenstrom mit ähnlichen Eigenschaften rekonstruiert werden.

Die Aufzeichnung und Wiedergabe von vollständigen Datenströmen inklusive der Protokoll-Nutzlast ist bei heutigen Netzen mit Durchsätzen, die teilweise oberhalb von 1Gbit sind, nur schwer möglich ist. Sowohl die Verarbeitungsgeschwindigkeit als auch die Speicherkapazität der Systeme, die den Datenstrom mitschneiden, reichen oft nicht aus, längere Testreihen durchzuführen.

Der offensichtliche Nachteil bei Verwendung von Traces ist die geringe Flexibilität. Beispielsweise kann eine veränderte Netzsituation (bezüglich Bandbreitenänderung oder Überlast) nicht eingearbeitet werden, da die zeitliche Abfolge der Pakete festgelegt ist. Insbesondere kann das Verhalten der Transportprotokolle (OSI Schicht 4 bzw. TCP) in Hinblick auf Flusskontrolle und Timeouts nicht berücksichtigt werden.

Eine verringerte Bandbreite oder andere Parameter des Verbindungsnetzes werden insofern berücksichtigt, dass die Dauer der Übertragung sich verlängert, weitergehende Effekte, wie beispielsweise Verbindungen, die auf Grund von Timeouts abgebrochen werden oder Pakete, die auf Grund von Übertragungsfehlern wiederholt werden müssen, können nicht nachgebildet werden.

Ein weiteres Problem des Trace-Verfahrens besteht darin, dass zwar ein exaktes Abbild des untersuchten Systems erstellt wird, aber die Netzlast als „black-box“ betrachtet wird [Barford, Crovella 1998]. Dies führt dazu, dass kein tieferes Verständnis der Zusammenhänge der jeweiligen Netzlast gewonnen werden kann. Außerdem ist ein Tracemodell auch wenig flexibel, und kann nicht auf Änderungen in den Last-Charakteristika angepasst werden, wie beispielsweise einer Veränderung der durchschnittlichen Dateigröße in einem Dateitransferprotokoll wie FTP.

Das Trace-Verfahren kann angewandt werden, wenn im Vordergrund der Untersuchung das Verhalten des Systems auf eine exakt vorgegebene Situation steht. Da es sich beim Trace-Verfahren um einen feststehenden Datenstrom handelt, kann exakt dieselbe Situation immer wieder simuliert werden.

3.2 Generierung eines aggregierten Datenstromes

Die Generierung von Last auf Netzwerkebene (IP im TCP/IP Modell) wird unter anderem von Hardware-Netzwerktestern wie beispielsweise den Smartbit-Geräten der Firma Spirent eingesetzt. Da beim Einsatz von Hardware-Netzwerktestern vor allem Messungen der Bitfehlerrate oder der Paketverlustrate eines Routers unter Volllast durchgeführt werden, ist ein realistisches Verhalten des Datenstromes bezüglich eines Anwendungsprotokolls wie beispielsweise HTTP nicht von Bedeutung. Bei der Erzeugung von Netzlast auf der Netzwerkebene werden wichtige Charakteristika einer echten Ende-zu-Ende Verbindungen nicht modelliert, wie beispielsweise das Slow-Start-Verhalten von TCP oder das erneute Senden beim Erreichen eines timeout, also wiederum das Verhalten des Transportprotokolls.

Ein weiteres Problem bei der Modellierung eines „realistischen“ Datenstromes besteht darin, dass sich die Ankunft von Netzwerkpaketen nicht mit der Poisson-Verteilung darstellen lässt. Die Poisson-Verteilung wird beispielsweise bei der Modellierung von Ankunftsrate oder Haltedauer von Telefongesprächen verwendet, und ist somit eine sehr oft eingesetzt und gut erforschte Verteilung. Bei der Beobachtung der Paketankünfte kann sie jedoch nicht eingesetzt werden. Durch die Abhängigkeit zu Paketen, die einige Zeit zuvor übertragen wurden (LRD = long range dependancy), besitzen die Pakete des Datenstromes schwierig nachzubildende Selbstähnlichkeitseigenschaften. Man spricht dabei von selbst-

ähnlichen oder fraktalen Verkehrsquellen. Ursache für dieses Verhalten ist nach [Taqqu 1997] die Überlagerung vieler Markovketten (Geburts-/Todesprozesse).

Insbesondere bei der Nachbildung großer Netze und hoher Bandbreiten ist der Vorteil bei der Erzeugung eines einzelnen aggregierten Datenstromes der geringere Aufwand im Vergleich zu Modellen, die Datenströme auf Anwendungsebene erzeugen. Eine denkbare Verwendung dieser Modelle ist vor allem die Erzeugung von Hintergrundlast.

Ein Beispiel für die Generierung eines aggregierten Datenstromes ist das Modell (M,P,S) [Lucas et al. 1997]. Dabei wird auf Basis von Messungen im Netzwerk der University of Virginia (UVANet) im Generator M ein selbstständlicher Datenstrom grob modelliert, indem pro 100 Millisekunden die Anzahl der zu sendenden Pakete bestimmt wird. Das Partitionierungsmodell P bestimmt dann die Aufteilung des von M erzeugten Datenstromes auf verschiedene Netze, ebenfalls basierend auf den Erkenntnissen aus der Messung im UVANet. Das "Short-term Packet Arrival Model" S bestimmt für jeden dieser Datenströme die Verteilung auf Millisekundenbasis nach dem Packet-train-Modell [Jain, Routhier 1986].

3.3 Generierung von Last nach dem white box modell

Bei der Erzeugung von Netzlast auf Anwendungsebene muss unterschieden werden zwischen der Verwendung der (gegebenenfalls modifizierten) Anwendungssoftware und der Erzeugung von Paketen mit den für die Anwendung charakteristischen Eigenschaften. Die Erzeugung von Netzlast durch die Verwendung der Anwendungssoftware erfolgt üblicherweise durch Steuerung der Clientsoftware durch Skripte oder Makros, oder durch abgeänderte Clientprogramme [JMeter]. Eine Möglichkeit, beispielsweise mit JMeter auf Anwendungsebene eine HTTP-Sitzung zu simulieren, besteht in einem HTTP-Client, der die Elemente einer Seite vom Server lädt und die weiterführenden Links analysiert. Nach einer durch das Anwendermodell festgelegten Pause, die das Lesen der Seite simuliert, wird zufällig ein Link ausgewählt und der Vorgang von Neuem gestartet. Der Einsatz von JMeter beschränkt sich im Wesentlichen auf HTTP und FTP Kommunikation.

Das white box Modell liefert die besten Ergebnisse, wenn sich wie bei einer WWW-Sitzung das Verhalten des Anwenders leicht modellieren lässt. Die durchschnittliche Verweildauer pro Seite und pro Server lässt sich aus den Protokollen von HTTP-Servern ermitteln.

Ein allgemein gehaltenes Beispiel für die Generierung von Last durch angepasste Anwendungsprogramme ist das Programm GenSyn [Heegaard 2000]. GenSyn ist ein Rahmenwerk zur Erstellung von whitebox Modellen. Die Modellierung des Anwenders erfolgt in Form einer Zustandsmaschine, die die unterschiedlichen Aktivitäten und den Übergang zwischen ihnen abbildet (siehe Abb. 6). Die Kommunikation selbst erfolgt in Form angepasster Client-Software, beispielsweise in Form eines HTTP-Client, der, ausgehend von vorgegebenen Adressen, Webseiten und deren Bilder lädt. Die in der Webseite enthaltenen Links können als Basis weiterer Übertragungen verwendet werden.

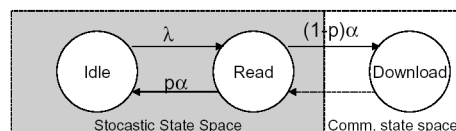


Abb. 6 GenSyn-Modell eines HTTP-Clients [Heegaard 2000]

GenSyn bietet Modelle für HTTP, FTP, Voice over IP, MPEG4 und Constant-Bit-Rate. In diesen wird die Aktivität des Anwenders auf mehreren Ebenen (Sitzungsebene, Seitenebene) simuliert, indem die Übergänge einer Zustandsmaschine mit den gemessenen Wahrscheinlichkeiten parametrisiert werden. Im Beispiel von Abb. 6 wird der Wert λ (Zeit zwischen

zwei Web-Sitzungen) negativ-exponentiell mit $1/\lambda = 1800$ Sekunden verteilt. Der Übergang zwischen dem „Stochastic State Space“ und dem „Comm State Space“ bedeutet dabei jedesmal den Auf- bzw. Abbau eines Kommunikationsinterface. Das Problem bei der Verwendung von white box Modellen besteht darin, dass individuell für jedes Protokoll eine Clientsoftware erarbeitet werden muss. Dies setzt gute Kenntnis der Abläufe des jeweiligen Protokolls voraus, da eine Interoperabilität mit „echten“ Servern sichergestellt sein muss, ist aber gleichzeitig auch Garant dafür, die individuellen Besonderheiten des Protokolls vollständig zu erfassen.

3.4 Generierung von Last nach statistischen Modellen von Anwendungsprotokollen

Die Erzeugung von Paketen mit anwendungscharakteristischen Eigenschaften hat den Vorteil, dass sich diese Methode flexibel auf verschiedene Anwendungen anpassen lässt. Das folgende Beispiel (Abb. 7) für die Erzeugung von Paketen, die den charakteristischen Eigenschaften einer TELNET Verbindung entsprechen, wurde [Danzig et al. 1992] entnommen. Die Werte für `telnet_duration`, `telnet_interarrival` und `telnet_pktsize` werden dabei entsprechend den statistischen Modellen generiert, die auf Grundlage von Messungen in verschiedenen Netzen erstellt wurden.

Der Lastgenerator „`tcplib`“ [Danzig, Jamin 1991] erzeugt Pakete nach Modellen für TELNET, FTP, NNTP und SMTP. Die Parameter der Modelle wurden auf Grundlage von Messungen in den Netzen der University of Southern California, Berkeley und Bell Communications Research erarbeitet. Allerdings findet bei `tcplib` keine echte Kommunikation statt, es ist vielmehr zur Lasterzeugung in einem Simulationsszenario gedacht.

Das Verfahren der Lasterzeugung nach statistischen Modellen auf Anwendungsebene ist das im weiteren Verlauf dieser Arbeit verwendete. Diese Entscheidung wurde vor allem in Hinblick auf die komplexen Probleme bei der Programmierung der einzelnen Clients beim Modell von GenSyn getroffen.

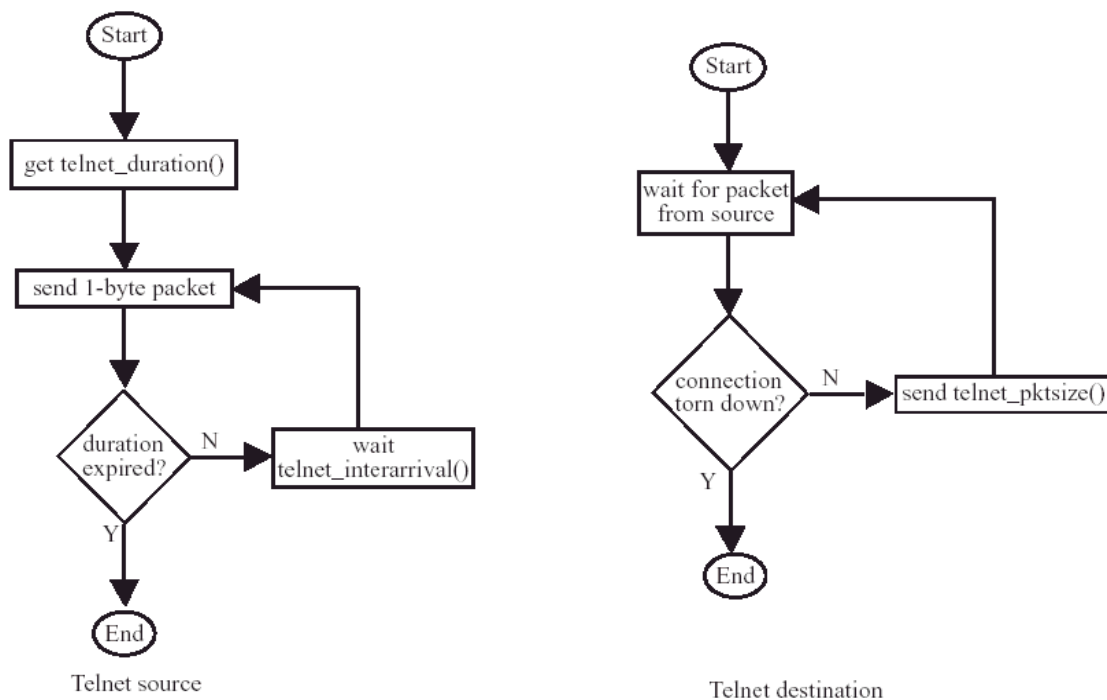


Abb. 7 Modell für Telnet [Danzig et al 1992]

Der Nachteil dieser Methode der Modellbildung besteht darin, dass sich Modelle für verschiedene Anwendungen oftmals nicht auf einfache Art gewinnen lassen [Barford, Crovella 1998]. Die Erstellung eines solchen Modells erfordert zunächst die Kenntnis der charakteristischen Größen einer Last. Diese sind bei verschiedenen Anwendungen völlig unterschiedlich, so spielt die Paketankunftsrate bei HTTP-Lastmodellen keine Rolle, bei Telnet-Lastmodellen ist sie dagegen die zentrale Charakteristik. Für die charakteristischen Größen eines Protokolls muss dann ein statistisches Modell gefunden werden. Schließlich muss ein zusammengesetztes Modell entwickelt werden, das die charakteristischen Größen eines Protokolls zu einem Gesamtmodell der Last verbindet.

3.5 Gegenüberstellung empirischer und analytischer Modelle

Die statistischen Modelle für die charakteristischen Größen eines Protokolls wie Verbindungsdauer oder Paketgröße können nach [Paxson 1994] in empirische und analytische Modelle eingeteilt werden. Ein empirisches (auch: heuristisches) Modell, wie in tcplib verwendet, modelliert die Verteilung der Zufallsvariablen entsprechend der zuvor gemessenen Verteilung in einem beobachteten Datenstrom.

Dahingegen versucht ein analytisches Modell [Paxson 1994], die im Datenstrom gemessenen Beobachtungen in einer geschlossenen Form darzustellen, also beispielsweise als log-normal Verteilung mit bestimmten Parametern für Mittelwert und Standardabweichung.

Der Vorteil des analytischen Modells besteht in der Skalierbarkeit. Während bei empirischen Modellen der Wertebereich durch die zuvor gemessenen Daten festgelegt ist, kann der Wertebereich bei analytischen Modellen beliebig erweitert werden, unter der Voraussetzung, dass das erarbeitete Modell auch für den erweiterten Wertebereich Gültigkeit besitzt. Ein weiterer Vorteil eines analytischen Modells besteht darin, dass es mit analytischen Modellen einfacher ist, mehrere Sachverhalte miteinander zu vergleichen.

Das wesentliche Problem bei der Erstellung analytischer Modelle besteht in der Frage, ob es überhaupt möglich ist, das Verhalten des Kommunikationssystems mit einer einfachen Formel nachzubilden. Oftmals besteht die Gefahr, mit der Vereinfachung der beobachteten Eigenschaften beispielsweise durch Glättung der Messwerte zur Annäherung an eine bestimmte Verteilung wesentliche Charakteristika des Systems zu vernachlässigen. Daher gibt die empirische Messung die tatsächliche Situation teilweise besser wieder, als ein angepasstes analytisches Modell.

Ein Beispiel für die Bedeutung von Spitzen in den Messdaten kann eine Paketgröße von 1492 Byte sein. Im TCP-IP Protokoll ist es für die Übertragung von großen Datenmengen sinnvoll, dies in den größtmöglichen Paketen zu tun. Dies reduziert das Verhältnis des Overhead in Form von TCP- und IP-Protokollköpfen zu den übertragenen Nutzdaten und verbessert so die Auslastung der Verbindung. Daher versucht ein Teilnehmer im TCP Protokoll immer die maximale Paketgröße zu verwenden, wenn genügend Daten zur Verfügung stehen. Die üblicherweise verwendete maximale Paketgröße (MTU = Maximum Transfer Unit) in lokalen Netzen beträgt 1500 Byte, da dies (zuzüglich der Paketköpfe) die maximal zulässige Paketgröße von Ethernet ist. Bei der Übertragung von Daten über eine DSL-Zugangsleitung (DSL = Digital Subscriber Line, digitale Teilnehmerleitung) werden die IP-Pakete jedoch üblicherweise im PPPoE Protokoll übertragen, das einen zusätzlichen Overhead von 8 Byte aufweist, so dass die MTU auf 1492 Byte verkleinert werden muss. Ist dies nicht bekannt, kann eine Glättung der Spitze bei 1492 Byte die analytische Verteilung der Paketgröße stark verfälschen, da sie diesen Wert falsch wiedergibt.

Allerdings gibt es auch Gegenbeispiele für die Bedeutung von einzelnen Werten. Bei der Analyse der mitgeschnittenen Daten in [Paxson 1994-2] wurde ein (im Vergleich zu Daten

anderer Messungen) sehr starkes Wachstum der FTP-Control Verbindungen beobachtet, die jede halbe Stunde Daten übertragen. Dies war auf ein Skript zurückzuführen, das in regelmäßigen Abständen Wetterkarten der Region übertrug. Da die Verkehrsmodelle im Allgemeinen solche Spezialfälle nicht berücksichtigen sollten, müssen die Messdaten zur Erstellung der analytischen Modelle gegebenenfalls bereinigt werden.

3.6 Ansätze zur Modellgewinnung

Nach [Mah 1997] können zur Gewinnung der Modelle für die charakteristischen Parameter von Kommunikationssystemen an drei Stellen Messungen durchgeführt werden: Am Server, am Client und im Verbindungsnetz zwischen beiden.

3.6.1 Protokollierung am Server

Der Vorteil der Protokollierung der Anwender-Aktionen am Server besteht im geringeren Aufwand im Vergleich zu einer Protokollierung auf Clientseite. Da ein Server von vielen Clients verwendet wird, ist es einfacher die Aktionen der Anwender zentral auf dem Server zu protokollieren, anstatt mit einer Protokollierung auf Clientseite die Aktionen aller Anwender einzeln erfassen zu müssen. Ein weiterer Vorteil besteht darin, dass viele Serveranwendungen ohnehin die Aktionen der Anwender protokollieren, so dass die Messungen möglicherweise ohne zusätzlichen Aufwand durchgeführt werden können.

Die wesentliche Beschränkung bei einer Protokollierung am Server liegt im fehlenden Wissen über das serverübergreifende Anwenderverhalten, da dies einen Abgleich der Protokolle aller in Frage kommenden Server erfordern würde.

Dies ist bei Anwendungen, bei denen der Server häufig gewechselt wird, wie beispielsweise bei HTTP, eine große Beschränkung, so dass dieses Messverfahren nur eingesetzt werden kann, wenn der Wechsel zwischen Servern keine charakteristische Größe des Protokolls darstellt.

3.6.2 Protokollierung am Client

Die clientseitige Protokollierung der Aktionen des Anwenders erfasst im Gegensatz zur Protokollierung am Server auch server-übergreifende Aktionen. Außerdem wird auch das clientseitige Zwischenspeichern von Daten (caching) erfasst, so dass auf diese Weise die Handlungen des Anwenders besser dokumentiert werden können. Ein Beispiel hierfür ist, dass während einer WWW-Sitzung von einem Webserver im Mittel zehn Dokumente gelesen werden [Catledge 1995], bevor auf einen anderen Webserver gewechselt wird. Übertragen werden jedoch nur vier bis fünf Dokumente [Mah 1997], da ein Rücksprung auf ein zuvor gelesenes Dokument keine erneute Übertragung des Dokuments erfordert, sondern das angeforderte Dokument aus dem lokalen Cache des Client gelesen werden kann. Im Gegensatz zu serverseitigen Protokollierung kann die clientseitige Protokollierung diese Tatsache erkennen.

Allerdings bringen diese Vorteile einen großen Aufwand mit sich. Die Client-Programme sind im Gegensatz zur Serversoftware in den seltensten Fällen in der Lage, die Aktionen zu protokollieren, so dass üblicherweise Anpassungen an der Clientsoftware notwendig sind. Dazu kommt, dass es deutlich einfacher ist, auf einem Server eine Protokollierungssoftware zu installieren, als auf vielen Client-Computern. Und nicht zuletzt haben Anwender oftmals die Befürchtung, dass sie durch die Messungen ausspioniert werden könnten,

selbst wenn eine anonyme Bearbeitung zugesichert wird, und werden sich daher möglicherweise weigern, die Software zu installieren.

3.6.3 Protokollierung der Datenpakete im Verbindungsnetz

Der Mitschnitt der Netzwerkpakete im Verbindungsnetz ist die zur Gewinnung von Verkehrsmodellen wohl am häufigsten eingesetzte Methode. Bei sinnvoller Platzierung der Messung, beispielsweise an einem zentralen Router eines Netzwerkes, können serverübergreifende Aktionen erkannt werden, und der Aufwand ist geringer als bei der clientbasierenden Methode.

Allerdings ist es nicht immer direkt möglich, aus den einzelnen Netzwerkpaketen, die zudem oftmals nicht als Ganzes, sondern lediglich mit ihren Paketköpfen protokolliert werden, auf das Verhalten auf Anwendungsebene zu schließen. Beispielsweise ist es nicht möglich, falls nur die Paketköpfe vorliegen, beim Mitschnitt einer HTTP-Datenübertragung zwischen Webserver und Browser Anfang und Ende (und damit auch die Größe) einer übertragenen Datei zu erkennen. Dies liegt daran, dass HTTP in der Version 1.1 TCP-Verbindungen wiederverwendet und daher mehrere Dateien über eine Verbindung übertragen werden. Eine Identifikation von Dateianfang- und Dateiende über Auf- und Abbau der TCP-Verbindung wie bei anderen Protokollen ist somit nicht möglich. Falls jedoch der Inhalt der Pakete auch protokolliert wird, können mit Hilfe von Wissen über die höheren Schichten der Kommunikation auch diese Daten festgestellt werden.

Wie bei der serverbasierenden Methode sind die Effekte des clientseitigen Zwischenspeicherns auch bei der Paketmitschnittmethode nicht direkt erkennbar. Allerdings ist bei der Gewinnung der Modelldaten das tatsächliche Verhalten der Anwender oftmals nur von zweitrangigem Interesse. Wenn statt dessen das Hauptaugenmerk auf den Auswirkungen des Verhaltens der Anwender, beispielsweise auf das Netzwerk, liegt muss dies keine Beschränkung darstellen.

4 Modelle verschiedener Anwendungen

Dieses Kapitel behandelt die Lastmodelle verschiedener Anwendungen, die in der Arbeit verwendet wurden.

4.1 Verwandte Arbeiten

Da die Erstellung der Anwendungsmodelle selbst nicht Gegenstand dieser Arbeit war, wurde auf bestehende Modelle anderer Arbeiten zurückgegriffen.

Ein auch in anderen Arbeiten oft zitiertes Werk, auf dem auch einige der in dieser Arbeit verwendeten Modelle basieren, ist der IEEE/ACM Artikel „Empirically Derived Analytic Models of Wide-Area TCP Connections“ von Vern Paxson [Paxson 1994]. Paxson erstellt anhand von Netzwerk-Mitschnitten aus verschiedenen US-Forschungseinrichtungen analytische Modelle für die charakteristischen Größen der Internetanwendungen FTP, SMTP und TELNET. Paxson verzichtet jedoch darauf, vollständig ausgeführte Gesamt-Modelle der Anwendungen zu entwickeln, sondern beschränkt sich darauf, die statistische Verteilung der einzelnen Verkehrsparameter zu entwickeln.

Eine ältere Arbeit ist „tcplib: A Library of TCP Internetwork Traffic Characteristics“ von Peter Danzig und Sugih Jamin [Danzig, Jamin 1991], in der heuristische Modelle unter anderem für FTP und TELNET erstellt wurden. Die Modelle sind auf Grund ihres Alters heute nur noch bedingt verwendbar, da sie die heutigen Werteverteilungen der Verkehrsparameter in der Regel nicht mehr widerspiegeln. Da sich aber bei der Benutzung von Telnet innerhalb der letzten 10 Jahre keine großen Veränderungen ergeben haben dürften, wird das TELNET-Modell von Danzig und Jamin in modifizierter Form in dieser Arbeit verwendet.

Das Modell für FTP-Last ist der Arbeit „FTP Traffic Generator“ von Joseph Ishac [Ishac 2001] entnommen. Ishac verwendet die analytischen Modelle aus [Paxson 1994] für die Parametrisierung seines Gesamt-Modells. Damit entsprechen die charakteristischen Größen zwar nicht den heutigen Werten, auf Grund der Verwendung analytischer Verteilungen können die Größen jedoch durch neue Messungen an heutige Verhältnisse angepasst werden.

Die Arbeit „An Empirical Model of HTTP Network Traffic“ von Bruce Mah [Mah 1997] beschäftigt sich mit der Erstellung eines Modells für HTTP-Last. Mah verwendet heuristische Verteilungen zur Angabe der charakteristischen Parameter von HTTP-Verbindungen. Dadurch ist eine Anpassung des Modells auf heutige Gegebenheiten nur bedingt möglich. Da das grundsätzliche Modell jedoch weiterhin anwendbar ist, und lediglich die Parameter wie beispielsweise die Anzahl der übertragenen Dateien auf dem Stand von 1997 sind, kann das Modell dennoch verwendet werden.

Für die Erstellung eines Modells von MPEG-Videoströmen wurde auf die Arbeit „Scene-based Characterization of VBR MPEG-Compressed Video Traffic“ von Marwan Krunz und Satish Tipathi [Krunz et al. 1996] zurückgegriffen. Darin wird ein Lastmodell entwickelt, das auf den Beobachtungen mehrerer digitalisierter Filme basiert.

Das Modell für Onlinespiele stammt aus den Arbeiten [Faerber 2002] und [Borella 1999], die den Netzwerkverkehr der Onlinespiele Counterstrike bzw. Quake mitschnitten und daraus Modelle für die Beschreibung der Verkehrsparameter entwickelten.

4.2 FTP

FTP ist ein Protokoll zur Übertragung von Dateien zwischen Rechnersystemen. Die interaktiven Aspekte des FTP-Protokolls, wie die Navigation durch einen Verzeichnisbaum, sollen an dieser Stelle nicht simuliert werden. FTP wird heute meist zur Übertragung von großen Datenmengen eingesetzt, so dass auf eine kurze interaktive Phase eine deutlich längere Phase der reinen Datenübertragung (bulktransfer) folgt. Außerdem sind die interaktiven Elemente wie Verzögerung der Kommandoeingabe weit weniger kritisch als bei wirklichen Dialoganwendungen wie beispielsweise Telnet. Zudem findet oftmals der interaktive Teil einer FTP-Sitzung, also das Navigieren durch eine Verzeichnisbaum, überhaupt nicht mehr statt, da Dateien, die per FTP übertragen werden, auch durch einen Link eines Dokuments im World Wide Web referenziert werden können. In diesem Fall findet das Navigieren über das HTTP-Protokoll statt und nur die eigentliche Dateiübertragung wird mit Hilfe des FTP-Protokolls bewerkstelligt.

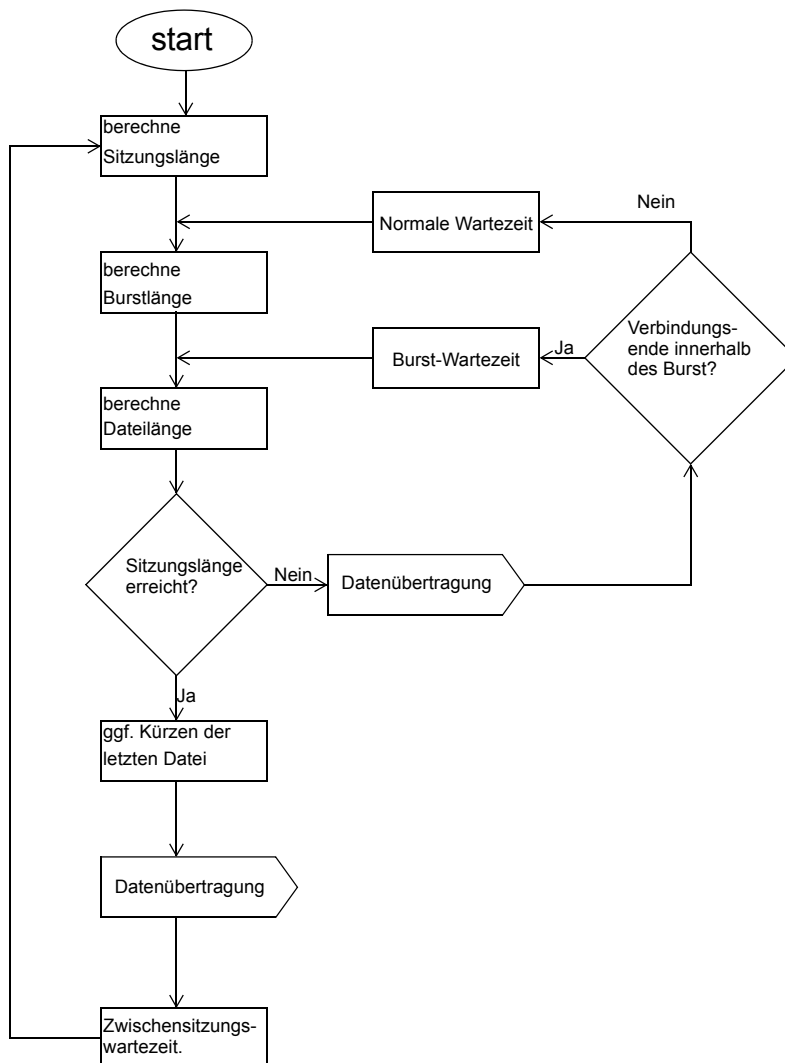


Abb. 8 Ablauf einer FTP-Verbindung

In [Ishac 2001] wird ein FTP-Netzlast-Generator auf Grundlage der Messwerte aus [Paxson 1994] beschrieben, der als Vorlage für die FTP-Lastgenerierung in dieser Arbeit dienen soll. Paxson entwickelte auf Basis von Messungen in den Netzwerken der Lawrence Berkeley Laboratories (Messreihen LBL1-8) und Digital's Western Research Lab (Messreihen DEC-WRL 1-4) sowie weiteren einzelnen Messungen Verteilungsmodelle für (unter anderem) FTP Kommunikation.

Messreihe	Pakete	Start der Messung	Ende der Messung
LBL-1	124 Mio	01.11.1990	01.12.1990
LBL-2	?	28.02.1991	30.03.1991
LBL-3	207 Mio	07.11.1991	07.12.1991
LBL-4	210 Mio	19.03.1992	18.04.1992
LBL-5	337 Mio	24.09.1992	23.10.1992
LBL-6	447 Mio	24.02.1993	26.03.1993
LBL-7	560 Mio	16.09.1993	15.10.1993
Bellcore	17k TCP-Verbindungen	10.10.1989	23.10.1989
UCB	38k TCP-Verbindungen	31.10.1989	01.11.1989
USC	5.2Mio	22.10.1991	23.10.1991
DEC-1	?	26.11.1991	27.11.1991
DEC-2	?	27.11.1991	28.11.1991
DEC-3	?	02.12.1991	03.12.1991
coNCert	63k TCP-Verbindungen	04.12.1991	05.12.1991
UK-US	26k TCP-Verbindungen	21.08.1991	21.08.1991

Verwendete Messwerte in den Arbeiten von Paxson [Paxson 1994, Paxson 1995]

Das darauf aufbauende Lastmodell von Ishac betrachtet den FTP-Datenstrom zuerst auf der Ebene des „Flows“, also der Menge der Sitzungen zwischen einem Client und einem Server. Innerhalb des Flow werden die Ebene der Sitzungen, und innerhalb dieser die Dateiübertragungen mit Hilfe von Paxsons Modellen beschrieben.

Im hier behandelten Lastgenerator wird die Zuordnung zwischen Client und Server sowie die Dauer der Lasterzeugung durch den Bediener des Lastgenerators festgelegt. Der Grund hierfür ist, dass der Lastgenerator für jedes angegebene Client-Server Paar die Sitzung eines Benutzers simuliert, nicht mehrere aggregierte Sitzungen zwischen Client und Server. Die Zeit zwischen zwei Sitzungen eines Benutzers liegen oftmals im Stundenbereich. Daher würde bei der Verwendung von wenigen Lasterzeugungs-Knoten (maximal 64 Knoten sind im Rahmen des Network Emulation Testbed vorgesehen) oftmals für sehr lange Zeiträume keine Last erzeugt werden, falls die Sitzungsebene auch modelliert werden würde.

Daher werden die Ebene des Flow sowie das Verhalten zwischen Sitzungen hier nicht weiter betrachtet, sondern lediglich der Datenstrom innerhalb einer Sitzung modelliert.

Weiterhin übernimmt Ishac aus Paxsons Arbeit den Begriff des Burst als eine Serie von Dateiübertragungen mit einer Pause von maximal 4 Sekunden zwischen den einzelnen Übertragungen. Bursts entstehen beispielsweise bei der Übertragung von Dateien mit dem Befehl „MGET“, bei dem mehrere Dateien zur Übertragung ausgewählt, und dann seriell übertragen werden.

Nach [Paxson 1994] kann die Größe der übertragenen Daten sowohl innerhalb der Sitzung als auch innerhalb einer einzelnen Dateiübertragung log2-normal-verteilt modelliert werden. Die Anzahl der Übertragungen innerhalb einer Sitzung wird dann ermittelt, indem so lange Dateiübertragungen generiert und deren errechnete Länge aufsummiert wird, bis die zuvor ermittelte Länge der Sitzung erreicht oder überschritten ist. Die letzte Übertragung muss daher gegebenenfalls gekürzt werden.

Die Parameter dieser log2-normal-Verteilungen sind stark von der zu simulierenden Umgebung abhängig. Beim FTP-Zugriff zum Aktualisieren eines Webserver werden üblicherweise viele kleine Dateien mit einer ebenfalls relativ kleinen Gesamtgröße übertragen, während bei einem Server, auf dem große Dateien wie CD-Images abgelegt sind, üblicherweise nur wenige sehr große Dateien übertragen werden.

Die Verteilung der Zeit zwischen zwei Übertragungen ist davon abhängig, ob sie als Burst übertragen werden, oder ob es sich um einzelne, voneinander unabhängige Übertragungen handelt.

Übertragungsart	Zeitraum	Bereich der CDF	Verteilung
Burst	0-2,5 Sekunden	[0..0,9095]	Exponentiell(0.616)
Burst	2,5-4 Sekunden]0,9095..1]	Uniform(2.5,4)
Normal	4-180 Sekunden	[0..1]	Log2-Normal ($\mu=3.27, \sigma=2.16$) +4 Sek.

Verteilung der Zeit zwischen zwei aufeinanderfolgenden Dateiübertragungen [Ishac 2001]

Bereich der CDF	Verteilung
[0..0,95]	Log2-Normal($\mu=10.3, \sigma=3.8$)
]0,95..1]	Pareto($\alpha=1, \beta=10^{5.5}$)

Verteilung der Burstlänge [Ishac 2001]

Um zu berechnen, ob mehrere Dateien in einem Burst übertragen werden, wird beginnend ab der ersten Übertragung die Burstlänge B berechnet. Alle darauffolgenden Übertragungen, die vollständig während der Burstlänge B übertragen werden, sind dem Burst zuzurechnen. Nach Übertragung einer Datei, die nicht innerhalb des Burst endet, wird eine neue Burstlänge berechnet.

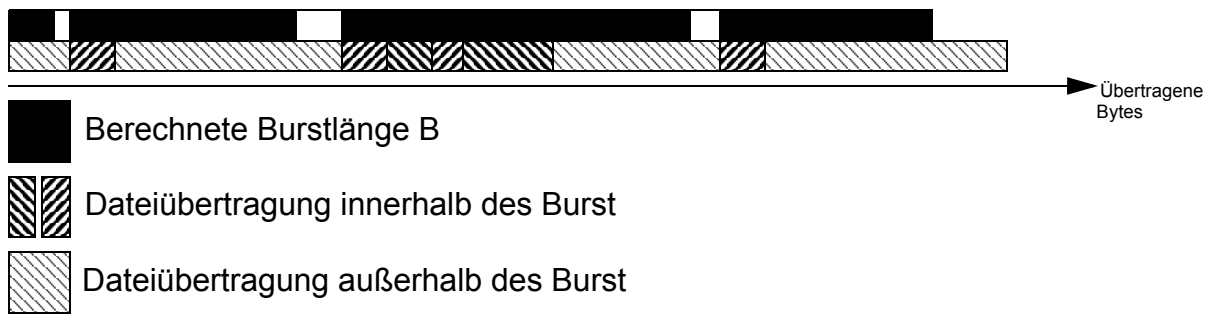
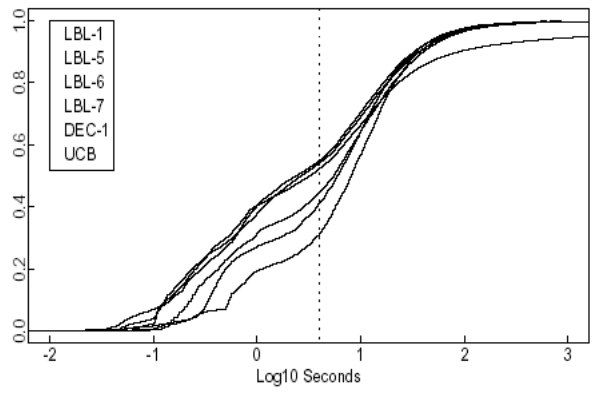
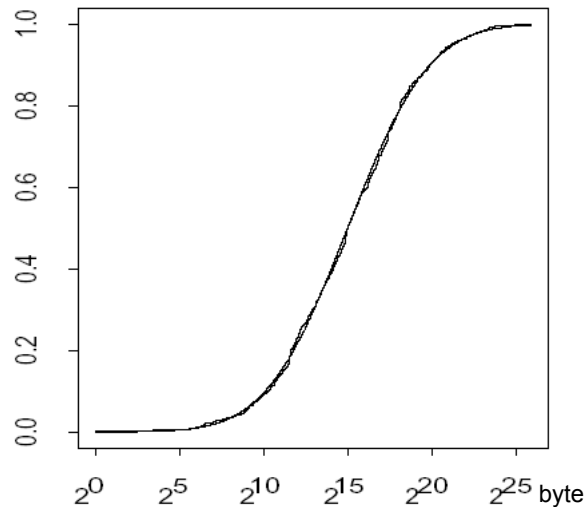


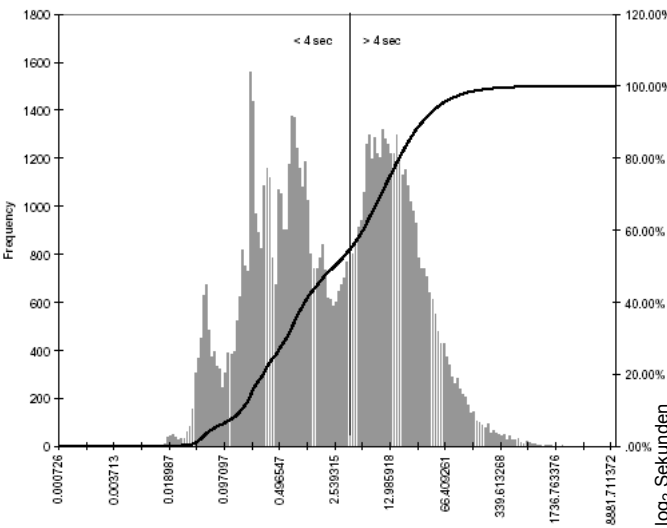
Abb. 9 Entscheidung der Burstzugehörigkeit



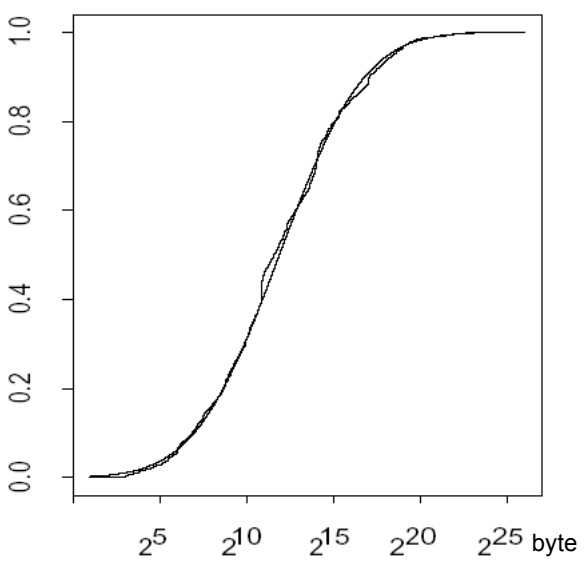
Zeit in Sekunden zwischen zwei FTP-Dateiübertragungen [1]



Verteilung der übertragenen Byte einer FTP Sitzung und eingepasste log2-normal Verteilung[2]



Zeit in Sekunden zwischen zwei FTP-Dateiübertragungen (log₂ transformierte LBL7 Messung)[3]



Verteilung der übertragenen Byte einer FTP Dateiübertragung und eingepasste log2-normal Verteilung[2]

[1] Vern Paxson, Sally Floyd: the failure of Poisson modeling
 [2] Vern Paxson: empirically derived analytic models
 [3] Joseph Ishac: FTP Traffic Generator

Abb. 10 Wahrscheinlichkeitsverteilungen des FTP Modells

4.3 HTTP

Die Verkehrscharakteristika des HTTP-Protokolls sind denen des FTP-Protokolls in vielen Aspekten sehr ähnlich. Wie auch das FTP-Protokoll wird das HTTP-Protokoll dazu verwendet, Dateien zu übertragen. HTTP wird als primäres Übertragungsprotokoll des World Wide Web (WWW) verwendet. Das World Wide Web ist ein verteiltes Hypertextsystem, somit ist ein zentrales Element, auch bei der Simulation des HTTP-Protokolls, die Nachbildung der Verweise zwischen den Dokumenten. Dokumente des WWW bestehen in der Regel aus einer HTML Datei und mehreren darin eingebetteten Objekte wie Bilder oder Mediendateien, die allerdings in der HTML-Datei nicht physikalisch eingebettet sind, sondern lediglich mit ihr logisch verknüpft sind. Das Wort „eingebettet“ beschreibt somit nur die letztendliche Wiedergabe der Objekte, da die Bilder in den Text der HTML-Datei eingebettet angezeigt werden.

In [Mah 1997] wird ein Modell für HTTP-Netzlast vorgestellt, das auf heuristischen Modellen auf der Grundlage von mitgeschnittenem Netzwerkverkehr im Netz der Berkeley Universität basiert. Mahs Modell unterscheidet zwei Arten von Verweisen. Einerseits werden die Verweise zwischen Dokumenten modelliert, indem nach Abruf einiger Dokumente der Server gewechselt wird. Zusätzlich werden die in die Dokumente „einbetteten“ Objekte modelliert, indem nach Abruf eines HTML-Dokuments weitere Dateien übertragen werden. Das HTTP-Protokoll öffnet in der ursprünglichen Version für jedes der zu übertragenden Objekte eine TCP Verbindung, die nach Übertragung des Objektes wieder geschlossen wird. Neuere Erweiterungen des HTTP-Protokolls wie „keep alive“ in HTTP 1.0 oder „persistent Connections“ in HTTP 1.1 halten die TCP Verbindungen nach Abschluss der Objektübertragung weiter offen, um bei weiteren Anfragen an denselben Server den beim TCP-Protokoll aufwändigen Verbindungsaufbau zu umgehen. Zudem werden üblicherweise mehrere Objekte die in einer Seite eingebettet sind, gleichzeitig übertragen, um den Seitenaufbau zu beschleunigen. Dies ist sinnvoll, da viele im WWW verwendeten Bildformate schon vor der vollständigen Übertragung als Vorschau angezeigt werden können. Die maximale Anzahl gleichzeitiger Übertragungen kann üblicherweise im Client (Browser) angegeben werden. Dies ist ein wesentlicher Unterschied zum FTP-Protokoll, bei dem im Allgemeinen die Dateien seriell übertragen werden, und somit eine TCP-Verbindung (zwei TCP-Verbindungen bei aktivem FTP) für ein Client-Serverpaar zur Datenübertragung ausreicht.

Zur Modellierung des HTTP-Protokolls werden die Werte für „Anfragegröße“, „Antwortgröße“, „Anzahl der Objekte eines Dokuments“ und „Zeit zwischen zwei Dokumentabrufen“ mit heuristischen Verteilungen nachgebildet. Die Werte für „Größe einer Anfrage“ und „Größe einer Antwort“ werden weiter unterschieden nach primären und sekundären Anfragen, also dem Abruf des HTML-Dokuments und den Abrufen der darin eingebetteten Objekte. Der Wert „Zeit zwischen zwei Dokumentenabrufen“ bezeichnet nicht die Zeit zwischen zwei primären Übertragungen, sondern die Zeit zwischen dem erfolgten Abruf des Dokuments inklusive aller eingebetteten Objekte und dem nächsten Abruf eines Dokumentes.

Der Wert „Zeit zwischen primärer und sekundärer Übertragung“, der in anderen HTTP-Modellen verwendet wird, wird dagegen nicht modelliert. Der Grund dafür ist, dass die Zeit, die eine moderne CPU benötigt, um ein HTML-Dokument zu interpretieren und die abzurufenden eingebetteten Objekte zu erkennen, im Vergleich zur Übertragungsdauer des primären Abrufs vernachlässigbar ist. Die Mehrzahl der heute verwendeten Browser beginnen zudem bereits während der Übertragung einer HTML-Datei mit deren Interpretation, so dass bereits während der Übertragung der HTML-Datei die eingebetteten Objekte identifi-

ziert werden können und mit ihrer Übertragung begonnen werden kann. Da über die Zeit zwischen Beginn der primären Übertragung und den sekundären Übertragungen, oder über die Anzahl der übertragenen Byte des primären Dokuments, bevor ein eingebettetes Objekt identifiziert und übertragen wird, keine Werte verfügbar sind, kann dieser Teil des HTTP-Modells nicht nachgebildet werden. Statt dessen wird direkt nach erfolgreicher Übertragung des primären Dokuments mit der Übertragung der eingebetteten Objekte begonnen.

Zusätzlich zu den oben genannten Werten werden zur Nachbildung der Wechsel zwischen den Servern die Werte „Dokumentenabrufe pro Server“ und „Serverauswahl“ benötigt. Im Gegensatz zu den anderen in dieser Arbeit verwendeten Modellen soll die Zuordnung von Client zu Server nicht fest sein, sondern es werden die bei einem Client angegebene Server abwechselnd befragt. Dies ist notwendig, da laut [Mah 1997] ein Client im Mittel lediglich ungefähr vier Dokumente von einem Server abrufen, bevor zu einem anderen Server gewechselt wird. Hält man sich die Grundidee des World Wide Web als verteiltes Hypertext-Systems vor Augen, so ist diese Tatsache leicht nachvollziehbar, da die Verknüpfung von Dokumenten auf verschiedenen Servern der Kernpunkt eines verteilten Hypertextsystems darstellt.

Die Anzahl der aufeinanderfolgenden von einem Server abgerufenen Dokumente, bevor zu einem anderen Server gewechselt wird, wird mit einer heuristischen Verteilung angegeben. Nach [Mah 1997] werden im Mittel zwischen 4 und 4,5 Dokumente von einem Server abgerufen, bevor zu einem anderen Server gewechselt wird. Diese Zahlen geben allerdings nur den Aspekt der Netzwerklast wieder, nicht den des wirklichen Anwenderverhaltens. In [Catledge 1995] wird die durchschnittliche Zahl der von einem Server angeforderten Dokumente mit 10 angegeben. Mah führt diesen Unterschied auf den Einfluss des Browsercaches zurück, der Dokumente für eine gewisse Zeit zwischenspeichert, so dass ein wiederholter Abruf eines Dokumentes innerhalb kurzer Zeit nur zu einer einzigen Übertragung über das Netzwerk führt. Ein Beispiel, in dem der Browsercache zu diesem Unterschied der gelesenen und übertragenen Seiten führen kann, ist ein Inhaltsverzeichnis, von dem zu einzelnen Kapitel verzweigt wird. Nach Lesen eines Kapitels wird wieder zum Verzeichnis gesprungen und zu einem anderen Kapitel verzweigt. Während bei [Catledge 1995] jeder Zugriff auf das Inhaltsverzeichnis als separater Zugriff gewertet wird, wird bei [Mah 1997] nur der erste Zugriff erfasst, da alle weiteren Zugriffe aus dem lokalen Browsercache bedient werden können. Da sich diese Arbeit nur mit den Netzlastaspekten des Benutzerverhaltens befasst, werden die Zahlen von [Mah 1997] verwendet.

Die Auswahl des nächsten zu befragenden Servers erfolgt nach dem Gesetz von Zipf [Almeida et al. 1996]. Wie die Poisson-Verteilung, die oftmals dann eingesetzt werden kann, wenn Ankunftsprozesse direkt von einem Anwender abhängig sind (Beispiel: Ankunft von Telefonaten, Haltezeit von Telefonaten), so ist die Zipfverteilung allgegenwärtig, wenn es um Fragestellungen der Auswahl eines Benutzers unter unterschiedlich beliebten Elementen geht. Die Zipfverteilung ist eine diskrete Verteilung und besagt, dass die Wahrscheinlichkeit, das Element aus einer Menge von Elementen mit unterschiedlicher „Beliebtheit“ mit dem i -ten Rang (es handelt sich also um das i -test beliebte Element) zu wählen, proportional zum Kehrwert des Ranges ist.

4.4 Digitales Video

Die Übertragung von digitalen Videostreamen über IP-Netze dürfte in Zukunft mit geplanten Dienstleistungen wie Video-On-Demand und interaktivem Fernsehen stark an Bedeutung gewinnen. Um die Verkehrscharakteristika von digitalen Videostreamen bestimmen zu können, muss zuerst der interne Aufbau von Videodateien betrachtet werden.

Unabhängig davon, ob ein Videostream nach MPEG2 (dem Kodierungsverfahren von DVDs und SVCDs) oder nach H.261 (dem Kodierungsverfahren für Bildtelefonie und Telekonferenzen) erzeugt wird, ist die grundsätzliche Struktur ähnlich. Eine Videodatei ist eine Serie von Bildgruppen (GOP=Group of Pictures), die wiederum aus mehreren unterschiedlich kodierten Rahmen (Frames) besteht. Zur zusätzlichen Ausnutzung von zeitlicher Redundanz neben der Räumlichen, die beispielsweise auch in der JPEG-Bildkompression angewandt wird, werden bei MPEG verschiedene Arten von Frames verwendet.

Ein I-Frame enthält ein vollständiges Bild, das mit der diskreten Cosinus Transformation, die auch in JPEG verwendet wird, kodiert ist. Ein P-Frame enthält dagegen nur die Unterschiede zwischen dem letzten I- oder P-Frame und dem aktuellen Bild. Filmsequenzen mit wenig Änderungen zwischen den einzelnen Bildern lassen sich so mit geringem Bandbreitenbedarf übermitteln. Die ebenfalls verwendeten B-Frames interpolieren zwischen dem vorhergehenden und dem folgenden I- oder P-Frame, um eine noch stärkere Datenkompression durch Ausnutzung zeitlicher Redundanz zu erreichen. Bei einer Kompression nach dem DivX Standard, der an MPEG4 angelehnt ist, werden P- und B-Frames als Deltaframes und I-Frames als Keyframes bezeichnet.

Bei DVD Qualität liegen die durchschnittlichen Rahmengrößen zwischen 50 und 100 Kilobyte [Agrawal et al. 1997] für einen I-Frame und 8 Kilobyte für einen kleinen B-Frame. Für Filme in VHS-ähnlicher Qualität (DivX-Kodierung) genügen 10 bis 15 Kilobyte für einen I-Frame und 4 bis 5 Kilobyte für einen Delta-Frame. Üblicherweise werden für eine Sekunde Film 30 Rahmen erzeugt.

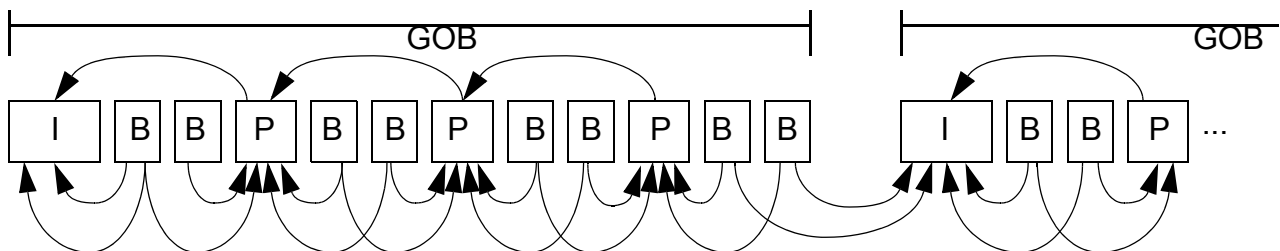


Abb. 12 Abhängigkeiten zwischen I-, P- und B-Frames eines MPEG-Stroms

Eine gebräuchliche GOP-Sequenz von MPEG2 (BBNC-Kodierer) lautet IBBPBBPBBPBB-PBB. Danach beginnt die nächste GOP-Sequenz mit dem folgenden I-Frame. Die GOP-Sequenzen können variieren. Beispielsweise ist es sinnvoll, falls bei einem Szenenwechsel innerhalb einer GOP die Größe der Delta-Frame stark ansteigt, mit dem Szenenwechsel eine neue GOP beginnen zu lassen. Analog kann bei Szenen mit wenig Bewegung die GOP-Sequenz auch deutlich verlängert werden.

Moderne Videokodierer wie DivX4, die vor allem auf schmalbandige Übertragung optimiert sind, verwenden variable GOP-Sequenzen und lediglich zwischen 0.5 und 1.5 Prozent der Frames sind Keyframes.

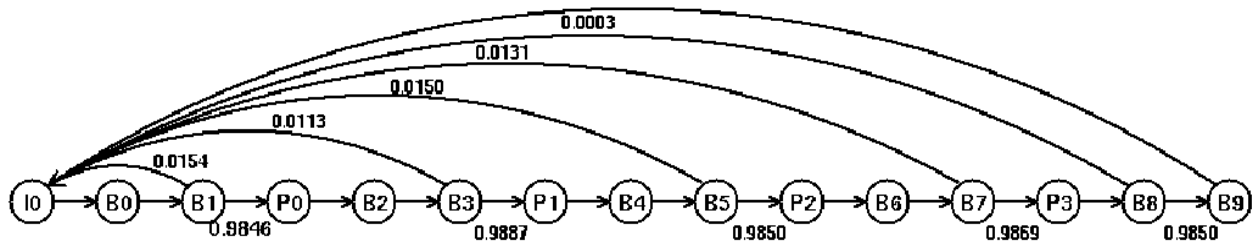


Abb. 13 Markov Modell für dynamische GOP Sequenzen [Agrawal et al. 1997]

Neben der Kompression hat die Wahl der GOP-Sequenz auch Auswirkungen auf die Fehleranfälligkeit der Videoströme. Bei Videofilmen ist eine Wiedergabe ohne Stockungen meist wichtiger als eine fehlerfreie Bildwiedergabe, wobei als Bildfehler beispielsweise eine falsche Farbe auf einigen Bildpunkten bezeichnet wird. Zum einen liegt dies daran, dass die Bildqualität durch die Kompression ohnehin schon vermindert wurde und vor allem bei der Übertragung von Videoströmen über langsame Netzwerke auch sichtbare Qualitätsverminderungen durch die Kompressionen in Kauf genommen werden. Somit fällt eine vorübergehende weitere Verminderung der Bildqualität durch ein fehlerhaft übertragenes Datenpaket möglicherweise nicht auf. Zum anderen wird eine Störung im zeitlichen Ablauf des Filmes im Allgemeinen subjektiv als störender wahrgenommen [Steinmetz 1996], als Fehler im Bild selbst. Daher werden Videoströme oftmals über UDP übertragen. Das UDP-Protokoll garantiert keine fehlerfreie Übertragung der Daten, muss daher aber auch keine Pakete wiederholt senden, was die Gefahr großer Varianzen in der Paketlaufzeit verringert. Fehlerhaft übertragene oder verlorengegangene Pakete bewirken dann Bildfehler, die in der Regel erst mit dem nächsten fehlerfrei übertragenen I-Frame behoben werden. Eine lange GOP-Sequenz bewirkt somit einen länger sichtbaren Bildfehler.

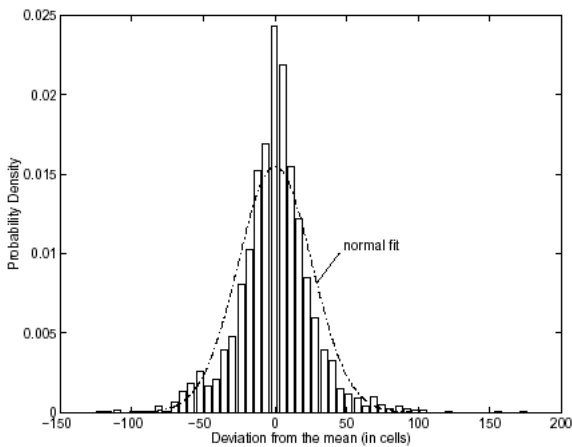
Bei der Simulation eines Videostreams müssen die Anwendungsfälle „Videokonferenz“ und „Spielfilm“ unterschieden werden. Bei ersterem treten im Normalfall keine Szenenwechsel auf und oftmals werden nur Deltaframes nach einem einzelnen Vollbild am Anfang der Übertragung übermittelt. Bei Videokonferenzen lässt sich daher ein synthetischer Datenstrom mit realistischen Eigenschaften dadurch erzeugen, dass mit der gewählten GOP-Sequenz I- und P-Frames erzeugt werden (H.263 kennt keine B-Frames).

Dagegen spielen die Szenenwechsel und die daraus resultierenden Veränderungen im Bandbreitenbedarf bei Spielfilmen eine große Rolle. Neben Modellierung der Verteilungen der I-,P- und B-Rahmengröße, die die Bildqualität des Filmes widerspiegeln, muss daher zusätzlich die Dynamik des Filmes modelliert werden. Im Folgenden sollen nur Spielfilme betrachtet werden.

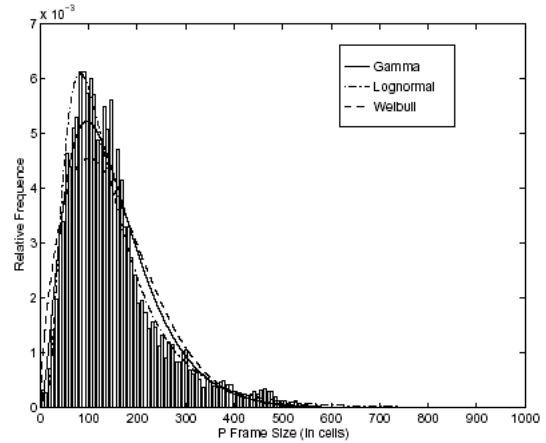
Das Modell für Videoströme in [Krunz et al. 1996] teilt den Datenstrom in Szenen ein. Eine Szene ist dabei als Folge von I-Frames, mit ähnlicher Größe definiert. Die Szenenlänge (angegeben als Anzahl von I-Frames innerhalb einer Szene) ist geometrisch verteilt mit der Dichtefunktion

$$P(n) = (1 - p)^{n-1} p$$

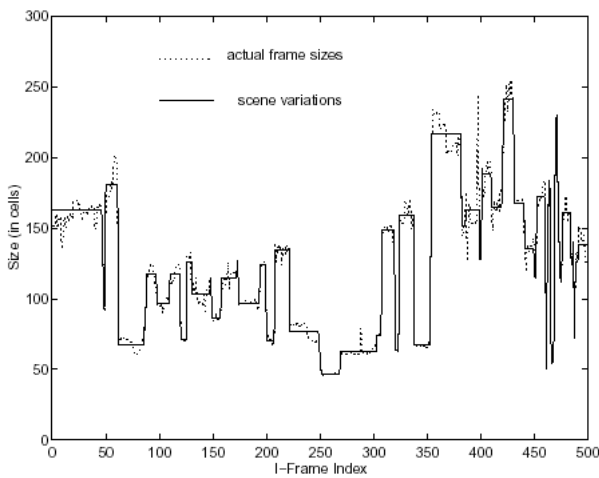
Damit steht $P(n)$ für die Wahrscheinlichkeit, dass die Szene aus n GOPs besteht. Der Verteilungsparameter p hängt von der Dynamik des Filmes ab. Ein hoher Wert für p weist auf einen Film mit kurzen Szenen hin.



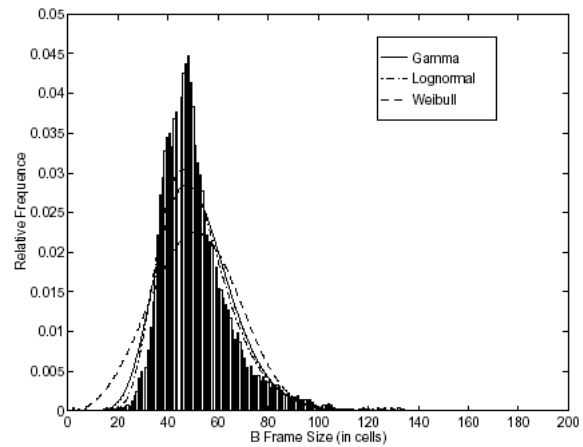
Abweichung der I-Framegröße vom Mittelwert innerhalb einer Szene (Herleitung von σ_ϵ)



Beobachtete Werteverteilung bei P-Frame Größen und Vergleich mit analytischen Verteilungsfunktionen



Ermittlung der Szenenlänge durch Analyse der I-Framegröße



Beobachtete Werteverteilung bei B-Frame Größen und Vergleich mit analytischen Verteilungsfunktionen

Abb. 14 Charakteristische Größen in Videos [Krunz et al. 1996]

Innerhalb einer Szene variiert die Größe der I-Frames definitionsgemäß wenig. Die Varianz zwischen I-Frames derselben Szene wird in [Krunz et al. 1996] mit der Autokorrelationsfunktion

$$\Delta(i) = a_1\Delta(i-1) + a_2\Delta(i-2) + \epsilon(i)$$

angegeben. Die Werte für $\epsilon(i)$ sind normalverteilt nach $N(0, \sigma_\epsilon^2)$.

Nach [Krunz et al. 1995] und [Krunz et al. 1996] kann die Größe von I-, B- und P-Frames mit Hilfe von lognormal-Verteilungen modelliert werden. Die Parameter der Verteilung sind von der gewünschten Filmqualität abhängig, [Krunz et al. 1996] nennt beispielsweise folgende Werte für die Filmsequenzen, die zur Herleitung des Modells verwendet wurden:

Datensatz	μ_I	σ_I	μ_P	σ_P	μ_B	σ_B
Wizard of Oz	5.9651	0.4832	4.8294	0.6443	3.9039	0.2746

Datensatz	μ_I	σ_I	μ_P	σ_P	μ_B	σ_B
Star Wars	4.9147	0.3861	3.9324	0.5842	2.8049	0.5242
Silence of the lambs	4.5813	0.3852	2.5884	0.8721	1.9861	0.6937
Goldfinger	5.3050	0.3663	4.5656	0.5111	3.2572	0.3761

Verteilungsparameter der Framegrößen [Krunz et al. 1996]

Die Werte beziehen sich dabei auf die Anzahl der benötigten ATM-Zellen mit 48 Byte Nutzlast. Die Dynamikparameter dieser Filme sind wie folgt angegeben:

Datensatz	a_1	a_2	σ_ϵ	ρ
Wizard of Oz	0.5300	0.1512	19.8	0.071
Star Wars	0.2088	0.0917	8.9644	0.17
Silence of the lambs	0.3965	0.1516	4.36	-
Goldfinger	0.4271	0.0380	10.3	-

Dynamikparameter der betrachteten Filme [Krunz et al. 1996]

Weiterhin verwendet der Generierungsalgorithmus die GOP-Sequenz (N,M), wobei N den Abstand zwischen zwei I-Frames angibt (und damit auch die Länge der GOP-Sequenz), und der Wert M den Abstand zwischen I- und folgendem P-Frame sowie den Abstand zwischen zwei P-Frames. Videoströme mit variierender GOP-Sequenz werden also nicht simuliert. Der Erzeugungsalgorithmus für Videoströme sieht dann wie folgt aus:

```

for j = 1 to Number_of_Scenes do
  Scene_Length = Ran_Genometric(p)
  I_Frame_Size Xi(j) = Ran_lognormal(muI, sigmaI)
  for i = 1 to Scene_Length do
    Delta(0)=0
    Delta(-1)=0
    for k = 1 to N do
      if Frametype(k) == I_Frame
        Delta(i) = a1*Delta(i-1)+a2*Delta(i-2)+
          Ran_Normal(0, sigmaepsilon2)
        Framesize == I_Frame_Size + Delta(i)
      if Frametype(k) == P_Frame
        Framesize = Ran_lognormal(muP, sigmaP)
      if Frametype(k) == B_Frame
        Framesize = Ran_lognormal(muB, sigmaB)
    endfor
  endfor
endfor

```

Eine Verbesserung in Hinblick auf variierende GOP-Sequenzen wird in [Agrawal et al. 1997] gezeigt. Anstatt eine konstante GOP-Sequenz zu verwenden wird dort mit einer 15-stufigen Markov-Kette gearbeitet. Die Übergänge zwischen den Zuständen sind mit den beobachteten Übergangswahrscheinlichkeiten in den Filmsequenzen, die zur Erstellung des Modells verwendet wurden, parametrisiert.

4.5 TELNET

Telnet ist eine Anwendung zum Absetzen von Kommandos an ein über ein Netzwerk mit dem Rechner des Anwenders verbundenes System. Üblicherweise wird dabei eine shell auf einem entfernten Rechner ferngesteuert. Dazu werden die Tastatureingaben im Telnet-Client des Anwenders über das Netzwerk zum Telnet-Server gesandt und dort in einer virtuellen Konsole ausgeführt. Im Normalbetrieb wird jedes eingegebene Zeichen vom Server als Echo zurück zum Client gesandt. Nach der Eingabe eines Kommandos wird dessen Ausgabe vom Telnet-Server zum Telnet-Client gesandt und dort angezeigt.

Damit zählt das Telnet-Protokoll zu den interaktiven Protokollen. Bei interaktiven Protokollen ist neben Parametern wie der Sitzungsdauer und der Anzahl der übertragenen Bytes auch die Paketankunftsrate von hoher Bedeutung. Dies rührt daher, dass aufgrund des niedrigen Bandbreitenbedarfs von Telnet die Paketankunftsrate weniger von den Parametern der Netzwerkes als vielmehr von der Tippgeschwindigkeit des Anwenders abhängig ist. Bei Bulktransfer-Protokollen lässt sich die Paketankunftsrate direkt aus der verfügbaren Bandbreite in Verbindung mit der Paketgröße berechnen, da die Daten im Allgemeinen so schnell wie möglich übertragen werden. Dagegen hängt die Paketankunftsrate bei Telnet nicht davon ab, wie schnell die Daten übertragen werden können, sondern wie schnell der Anwender sie produzieren kann.

Eine Telnetsitzung besteht im wesentlichen aus zwei Arten von Paketen: Jeder Tastendruck des Anwenders wird als Ein-Byte-Paket zum Server übertragen und die Änderung der Server-Konsole (ebenfalls ein Ein-Byte-Paket) vom Server zum Client. Nach erfolgter Eingabe des Befehls wird die Ausgabe des dadurch gestarteten Programms (in möglichst großen Paketen) vom Server zum Client gesandt. Die modellspezifischen Größen sind damit die Ankunftsrate der Pakete vom Client zum Server, die Größe der Antwort vom Server zum Client und die Länge eines Kommandos. Wie auch beim FTP-Protokoll wird die Sitzungsdauer nicht nachgebildet, um die langen Zeiten ohne Netzlast zwischen zwei Sitzungen zu vermeiden.

Der Wert „Antwortgröße“ beschreibt die Größe der Ausgabe des abgesetzten Kommandos. In [Paxson 1994] wird sie als das Verhältnis der vom Server gesendeten Bytes zu den vom Client gesendeten Bytes mit einer \log_2 -normal Verteilung mit $\mu = \log_2 21$ und $\sigma = \log_2 3.6$ angegeben. Dies schließt aber die bereits gesendeten Echo-Pakete mit ein, so dass sich die tatsächliche Antwortgröße zu

$$Anwortbytes = 2^{Normal((\log_2 21 - 1) \times empfangeneBytes; \log_2 3, 6)}$$

ergibt.

Die Verteilung der Ankunftsrate modelliert die Paketankünfte der Ein-Byte-Pakete des Benutzers, und damit dessen Tippgeschwindigkeit. Obwohl diese Pakete direkt durch den Benutzer erzeugt werden, handelt es sich bei der Verteilung der Ankunftsereignisse dennoch nicht um die Poisson-Verteilung. Diese kann normalerweise dann verwendet werden, wenn ein Ereignis direkt von einem Anwender abhängig ist, wie beispielsweise Ankunftsrate und Haltezeit von Telefonaten, oder Ankunftsrate von FTP-Sitzungen. Eine Modellie-

Die Schätzung der Paketankunftsrate des Anwenders als Poisson-Verteilung unterschätzt nach [Paxson 1995] die „burstiness“, also die Ankunft der Paketen in Schüben oder „Büschel“, die unter anderem durch die unterschiedliche Paketverzögerung verursacht wird.

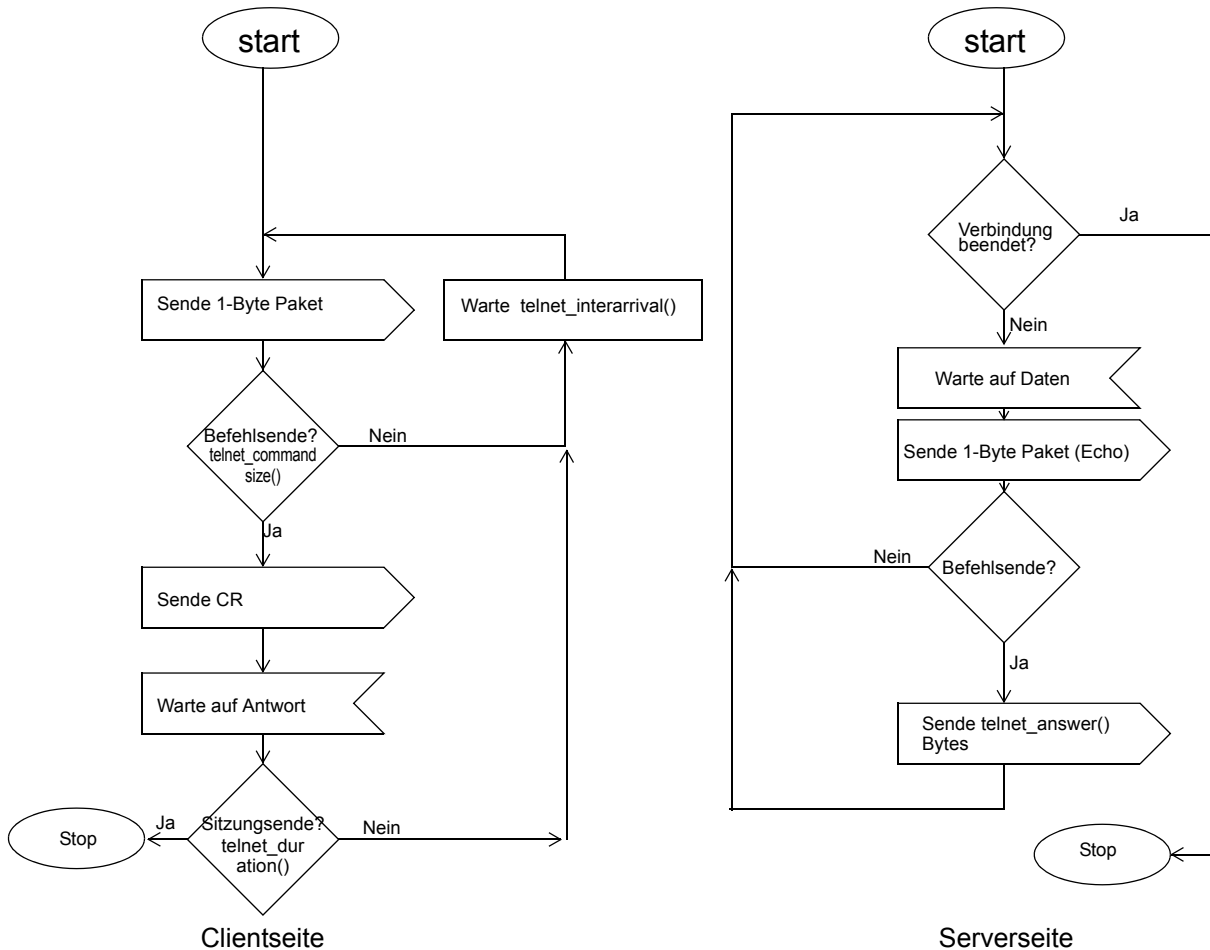


Abb. 15 Ablauf einer Telnet-Verbindung

In [Paxson 1995] wird die Zeit zwischen zwei Paketen mit zwei Pareto-Verteilungen, mit $\beta=0.9$ beziehungsweise $\beta=0.95$ für die oberen 3% modelliert. Hier soll statt dessen die heuristische Verteilung von [Danzig et al 1992] verwendet werden, die auch im Lastgenerator „tcplib“ verwendet wurde. Diese Verteilung wurde anhand von über zwölf Millionen mitgeschnittenen TCP-Paketen aus Netzen zweier Universitäten und der Forschungseinrichtung Bellcore ermittelt. Diese Messungen fanden zwar zwischen 1989 und 1991 statt, da sich am üblichen Nutzungsverhalten von telnet-Verbindungen aber wenig geändert hat, dürften diese Werte immer noch Gültigkeit besitzen.

Die Befehlslänge beschreibt die Länge einer Eingabe, also die Anzahl der 1-byte Pakete, die vom Client zum Server übertragen werden, bis das Kommando beendet ist und die Antwort des Programms gesendet wird. Da keine verwertbaren Informationen zur Verteilung von Kommandolängen auf Unix-Kommandozeilen zu finden waren, wird diese Verteilung anhand von selbst erstellten Heuristiken (Abb. 16) festgelegt.

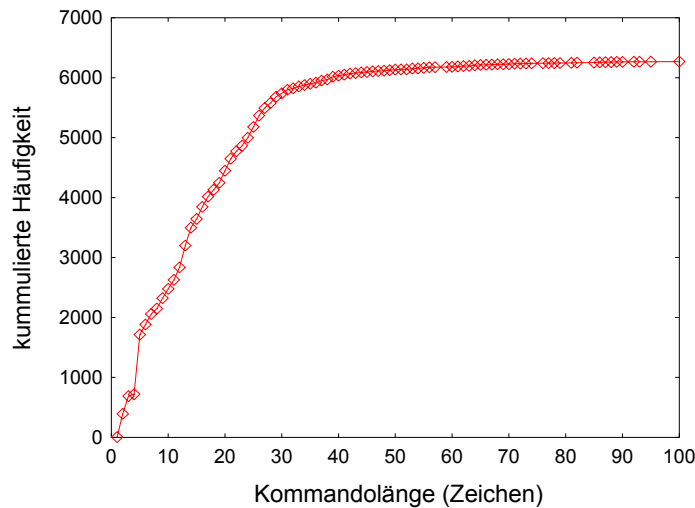


Abb. 16 Länge von Kommandos in Telnetsitzungen

4.6 Onlinespiele

Daten von Online-Computerspielen sind ein in letzter Zeit nicht mehr vernachlässigbarer Teil der Netzwerkverkehrszusammensetzung. Insbesondere die Ausrüstung neuer Spielekonsolen mit Netzwerkanschlüssen lässt ein auch künftiges Anwachsen des durch Spiele verursachten Netzwerkverkehrs vermuten.

Wie auch bei Telnet handelt es sich bei Computerspielen um interaktive Anwendungen, so dass nicht die Bandbreite, sondern die Paketverzögerung der für den Anwender wichtigste Parameter des Verbindungsnetzwerkes ist. Insbesondere bei actionlastigen Onlinespielen (First Person Shooter) ist der Einfluss der Paketverzögerung auf die Spielbarkeit noch höher als bei typischen Dialoganwendungen wie telnet, aber auch einer Videokonferenz oder der (Sprach)-Telefonie. Während bei letzterer die Dialogfähigkeit bis zu einer Verzögerung (Hin- und Rückweg) von 500 ms gegeben ist, sind Onlinespiele bei einer Verzögerung jenseits 100 bis 150 ms schon kaum mehr spielbar [Borella 1999].

Onlinespiele verwenden als grundsätzliche Verbindungsarchitektur entweder eine Vollvermaschung, bei der jeder Teilnehmer mit jedem kommuniziert, oder eine Client-Server Struktur, bei der jeder Client nur mit dem Server kommuniziert. Aufgrund der oftmals unbestimmten Anzahl der Mitspieler - Spiele wie Ultima Online verwalten mehrere Tausend gleichzeitige Mitspieler - ist bei den meisten Spielen ein Client-Server Modell implementiert. Der bei Quake und Counterstrike beobachtete Verarbeitungszyklus [Borella 1999] beginnt mit dem Senden von Zustandsinformationen der Spielwelt vom Server an alle Clients. Nach dem Verarbeiten der Zustandsinformationen wird von jedem Client ein Updatepaket an den Server gesandt, in dem vor allem die Tastatur- und Mauseingaben des Spielers an den Server übermittelt werden. Dieser berechnet die aus den Eingaben der Spieler resultierende Auswirkungen auf die Spielwelt und sendet diese in Form neuer Zustandsinformationen an alle Clients.

Da Multicast-Protokolle, die für die Übertragung der üblicherweise für alle Client-Rechner identischen Zustandsinformationen eigentlich ideal wären, noch in den seltensten Fällen vorausgesetzt werden können, wird die Kommunikation (in beide Richtungen) üblicherweise über UDP-Datagramme abgewickelt. Bei den hier behandelten First Person Shootern ist die Paketverzögerung von sehr hoher Bedeutung, da es sich um sehr schnelle Spiele handelt. Würde anstatt einer UDP- eine TCP-Verbindung verwendet, würden die

Datenpakete oftmals mit hoher Verzögerung übertragen werden. Dies würde in einem stotternden Spielverlauf resultieren. Daher wird das verbindungslose UDP verwendet, da ein wiederholtes Paket ohnehin nicht mehr verwendet werden kann. Daten die zuverlässig übermittelt werden müssen, können durch Sicherungsmaßnahmen auf der Anwendungsschicht (beispielsweise durch bestätigte Datagramme) übermittelt werden, ohne dass der sonstige Spielverlauf gestört wird [Bragenza, Pamidi 2000].



Abb. 17 Ablauf der Onlinespiele-Kommunikation

Der interaktiven Phase der Kommunikation geht vielen Spielen eine Phase der Datenübertragung voraus, bei der die Version des Programms und der Spielumgebung zwischen Server und den einzelnen Clients abgeglichen wird. Diese Datenübertragung soll jedoch im Folgenden nicht berücksichtigt werden, da keine Daten über deren Charakteristika vorliegen.

In [Faerber 2002] und [Borella 1999] werden als alleinige Verkehrscharakteristika die Paketankunftsrate zwischen Client und Server beziehungsweise zwischen Server und Client, sowie die jeweiligen Paketgrößen identifiziert. Beide Arbeiten bilden die Paketgröße der Serverpakete sowie die Paketankunftsrate der vom Server an alle Clients gesendeten Pakete mit einer Extremal-Verteilung nach. Während bei [Borella 1999] die Paketankunftsrate der Clientpakete ebenfalls mit einer Extremal-Verteilung nachgebildet wird, verwendet [Farber 2002] eine konstante Paketankunftsrate für Clientpakete. Die Größe der Clientpakete wird bei [Borella 1999] mit einem konstanten Wert von 24 Byte angegeben, während [Faerber 2002] eine Extremal-Verteilung verwendet.

Wert	Counterstrike	Quake	Quake 2
Client Paketankunftsrate in ms	konstant 40[Extremal ($x_0=40, b=0$)]	Extremal ($x_0=23.06, b=5.23$)	Extremal ($x_0=30.26, b=8.01$)
Client Paketgröße in Byte	Extremal ($x_0=40.0, b=5.7$)	konstant 24 [Extremal ($x_0=24, b=0$)]	konstant 24 [Extremal ($x_0=24, b=0$)]
Server Paketankunftsrate in ms	Extremal ($x_0=55.0, b=6.0$)	Extremal ($x_0=44.20, b=9.52$)	Extremal ($x_0=22.84, b=5.29$)
Server Paketgröße in Byte	Extremal ($x_0=120.0, b=36.0$)	Extremal ($x_0=89.92, b=34.84$)	Extremal ($x_0=65.58, b=19.12$)

Verkehrsparameter in Onlinespielen [Farber 2002, Borella 1999]

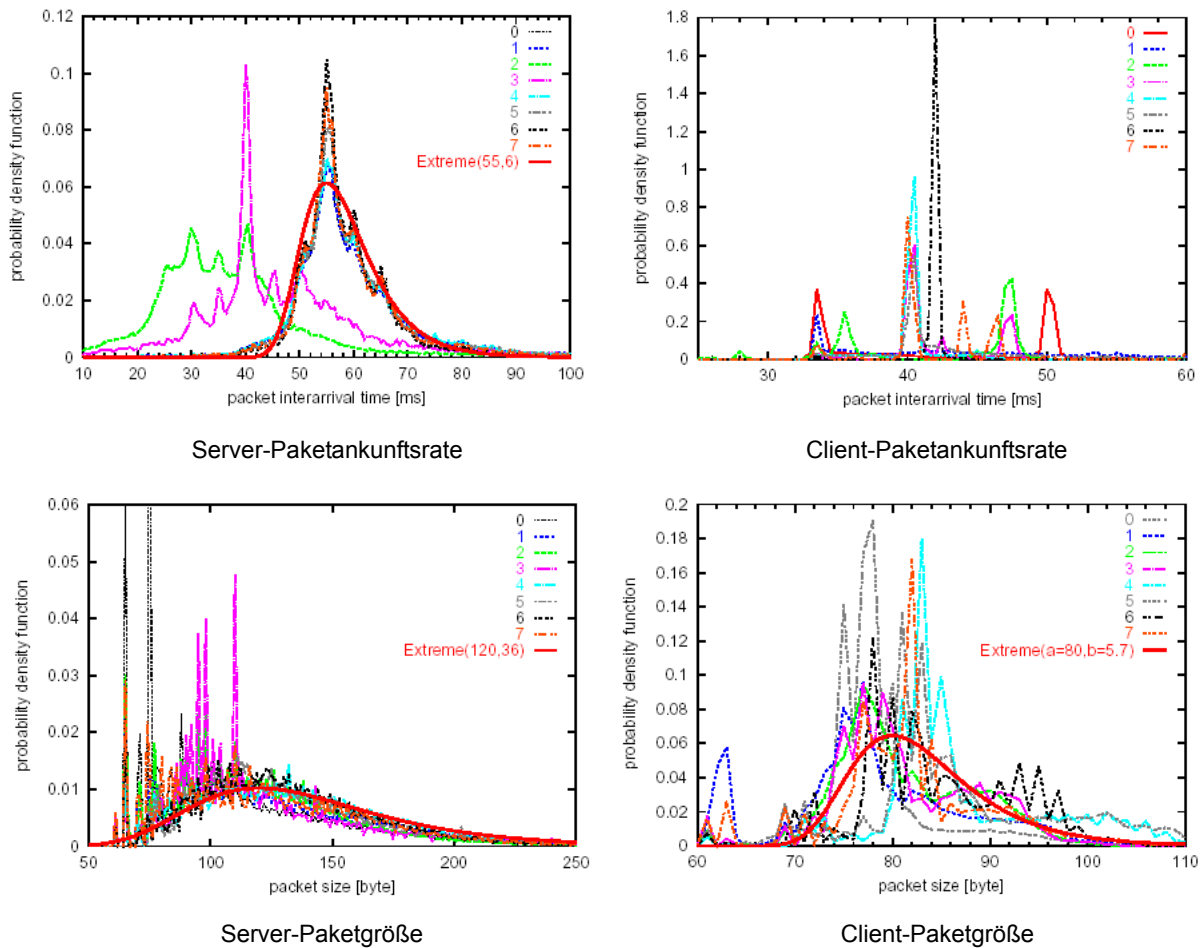


Abb. 18 Statistische Verteilungen in Counterstrike [Faerber 2002]

Eine deterministische Verteilung, wie sie beispielsweise bei der Quake-Clientpaketgröße verwendet werden muss, kann mit einer Extremal-Verteilung mit dem gewünschten konstanten Wert für den Ortsparameter x_0 und dem Formparameter $b=0$ modelliert werden kann. Daher kann für alle drei Onlinespiele auf dasselbe Lastmodul zurückgegriffen werden. Lediglich die Parameter sind entsprechend den Messungen aus den Arbeiten von Borella und Färber anzupassen.

Die Diagramme in Abb. 18 zeigen die gemessenen Verteilungen der Größen Paketankunftsrate und Paketgröße auf Client und Serverseite, sowie die eingepassten Extremal-Verteilungen zur möglichst guten Wiedergabe der gemessenen Werte.

Die in der Tabelle genannten Parameter der Extremal-Verteilung der Quake-Clientpaketankunftsrate sind Beispiele und nicht der Mittelwert für alle Clients. Da Borella lediglich drei beziehungsweise bei Quake 2 vier Clients untersuchte, und die Paketankunftsrate der Clientpakete stark von der Aktivität des jeweiligen Spielers abhängig ist, können keine Aussagen über die durchschnittlichen Parameter der Extremal-Verteilung gewonnen werden.

Die Angaben zur Ankunftsrate der Serverpakete beziehen sich jeweils auf eine Client-Server Verbindung. Die Verzögerung zwischen Paketen derselben Spiel-Änderung, die an die Clients übertragen wird, ist nach [Borella 1999] nahezu 0 ms, so dass diese Pakete praktisch ohne Verzögerung übertragen werden. Das Last-Modell für Onlinespiele ist in Abb. 18 dargestellt.

5 Architektur der Software

Die Lastgenerierungs-Software ALTGen (Application Level Traffic Generator) besteht aus drei Teilen: der grafischen Benutzungsoberfläche, der Knotensteuerung, sowie verschiedener Module zur Erzeugung von Netzlast.

Als Programmiersprache für den Lastgenerator wird Java verwendet. Damit wird es möglich sein, den Lastgenerator ohne Änderungen auf verschiedenen Plattformen zu verwenden, beispielsweise auch unter Emulationsszenarien mit gemischten Betriebssystemen.

5.1 Benutzungsoberfläche (GUI)

Die GUI bietet ein Rahmenwerk zur Integration der Konfigurationsschnittstellen der einzelnen Module sowie zur Kommunikation mit der Knotensteuerung auf den einzelnen Knoten des NET-Clusters. Die Verbindung zwischen Elementen der Konfigurationsoberfläche der Module und den zugrundeliegenden Daten erfolgt nach dem für java-Swing üblichen Model-View-Controller (MVC) Konzept. Dazu werden drei Komponenten unterschieden:

1. Das Datenmodell „Model“ beinhaltet die interne Repräsentation des aktuellen Zustands, also beispielsweise einen String.
2. Die Ausgabeschnittstelle „View“ beinhaltet die visuelle Repräsentation des internen Zustands, wie beispielsweise eine Textausgabe.
3. Die Eingabeschnittstelle „Controller“ bietet eine Möglichkeit, den internen Zustand zu ändern. Ein Controller kann beispielsweise eine Texteingabe sein.

Die Komponenten stehen in folgender Wechselwirkung:

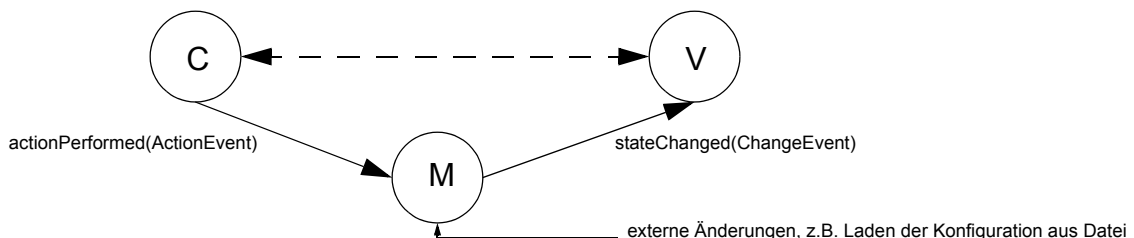


Abb. 19 Das MVC-Konzept

Die Verbindung der Komponenten erfolgt unter java-Swing mittels Callbacks. Beispielsweise registriert ein „View“ beim „Model“ einen ChangeListener-Callback zu sich selbst, der vom „Model“ aufgerufen wird, sobald die Daten des „Model“ geändert wurden. Analog wird ein ActionListener-Callback des „Model“ beim „Controller“ registriert, den dieser bei Eingaben bzw. Änderungen des Anwenders aufruft.

„Controller“ und „View“ können sich auf dieselbe GUI-Komponente beziehen, beispielsweise kann ein Texteingabefeld vom Typ JTextField sowohl die visuelle Repräsentation eines Strings des Model sein, gleichzeitig aber auch eine Eingabemöglichkeit zum Ändern des Datums bieten.

Ein „Model“ kann mehrere „Controller“ und „Views“ besitzen. Beispielsweise kann eine Temperatursteuerung Eingaben in Grad Celsius und Grad Fahrenheit zulassen, die sich beide auf dasselbe „Model“ mit einem Float-Wert „Temperatur in Kelvin“ beziehen. Zusätzlich kann noch eine grafische Temperaturanzeige als „View“ registriert werden, die nicht gleichzeitig ein „Controller“ ist.

5.2 Knotensteuerung

Die Knotensteuerung dient als zentrale Kommunikationskomponente zwischen den Lastmodulen und der GUI. Dazu verfügt sie über eine Kommunikationsschnittstelle zum Empfang der Konfigurationsdaten von der GUI, sowie über Methoden zur Steuerung der Module.

Jeder Knoten des Emulationssystems verfügt über eine eigene Knotensteuerung, die für die Kontrolle und Konfiguration der Lastmodule dieses Knoten zuständig ist.

5.3 Lasterzeugungs-Module

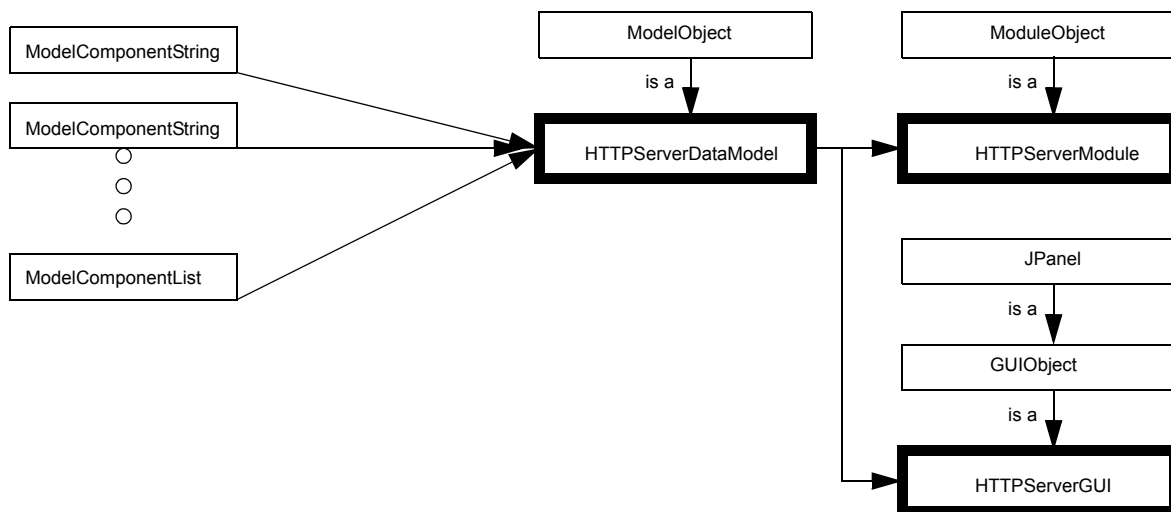


Abb. 20 Architektur eines Lasterzeugungs-Moduls

Jedes Lasterzeugungs-Modul besteht aus drei Teilen:

1. Das Datenmodell ist die Menge der Konfigurationseigenschaften des Moduls. Es besteht aus mehreren Objekten der Klassen „ModelComponent[String|List]“, die eine Modelvariable (z.B. Liste der Server mit denen ein Client-Modul kommuniziert, oder die mittlere Sitzungslänge) kapselt, und Schnittstellen für die Registrierung von „Controller“ und „View“ nach dem MVC-Modell zur Verfügung stellt.
2. Die grafische Konfigurationsschnittstelle wird in die grafische Benutzungsoberfläche des Hauptprogramms eingebunden. Über die Konfigurationsschnittstelle lassen sich die einzelnen Komponenten des Datenmodells modifizieren. Dazu werden die Komponenten des Datenmodells mit GUI-Objekten wie Listen und Eingabefeldern mittels des Model-View-Controller - Modells verbunden.
3. Das Lastmodul wird von der Knotensteuerung gesteuert und erzeugt gemäß der übermittelten Konfiguration Datenpakete.

6 Implementierung der Statistikfunktionen

Für die Generierung von Netzlast ist es notwendig, zufällige Ereignisse nach vorgegebener Verteilung zu generieren. Allerdings bietet die mit Java ausgelieferte Bibliothek lediglich die Möglichkeit, Zufallszahlen mit uniformer Verteilung und Normalverteilung zu generieren. Zur Generierung von Zufallszahlen mit vorgegebener nicht-uniformer Verteilung lassen sich für alle Dichtefunktionen numerische Methoden anwenden. Da die Zufallszahlen für die Erzeugung von Netzlast jedoch sehr häufig benötigt werden - teilweise sind für jedes Paket mehrere zufällige Werte zu generieren, um dessen Größe oder die genaue Ankunft zu berechnen - ist es unabdingbar, für häufig verwendete Verteilungen schnellere Generierungsvorschriften zu verwenden. Diese sind dann allerdings oftmals nur für eine bestimmte Verteilung anwendbar. Für die Erstellungen der Modelle nach Kapitel 4 werden folgende Verteilungen benötigt: (log)-normal, Pareto, (log)-Gumbel, Exponential, Poisson, Zipf, Weibull, geometrische und heuristische Verteilungen. Eine Möglichkeit, mit geringem Aufwand nicht-uniform verteilte Zufallszahlen zu erzeugen, besteht in der inversen Transformation.

6.1 Inverse Transformation (exponential-, Pareto- und Gumbel-Verteilung)

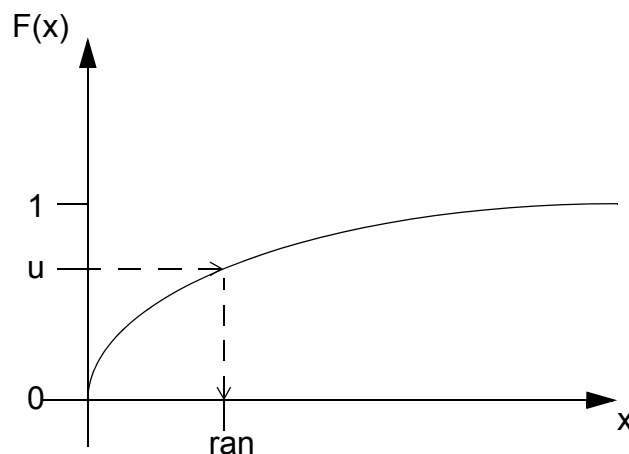


Abb. 21 Methode der Inversen Transformation

Um nach dem Verfahren der inversen Transformation eine Zufallszahl *ran* nach einer vorgegebenen Verteilung zu erzeugen, wird zuerst eine Zufallszahl *u* zwischen 0 und 1 nach der uniformen Verteilung erzeugt.

Zur Berechnung der gewünschten Zufallszahl *ran* mit Hilfe von *u* wird die inverse Funktion $F^{-1}(x)$ der Verteilungsfunktion benötigt. Die gewünschte Zufallszahl *ran* ergibt sich dann als $ran = F^{-1}(u)$ [Devroye 1996].

Offensichtlich ist die Kenntnis der Inversen der Verteilungsfunktion Voraussetzung, um eine Zufallszahl nach der Methode der inversen Transformation zu erzeugen. Die direkt berechenbaren Inversen von in dieser Arbeit verwendeten Verteilungen sind im folgenden aufgeführt.

Verteilung	inverse Verteilungsfunktion
Exponential	$F^{-1} = -\frac{\ln(1-u)}{\lambda}$
Pareto	$F^{-1} = \frac{\beta}{u^{\frac{1}{\alpha}}}$
Extremal (Gumbel)	$F(x) = x_0 - b \ln(-\ln(u))$

6.2 Annäherung einer geschlossenen Funktion (Normal-Verteilung)

Viele der anderen Verteilungsfunktionen können nicht in einer geschlossenen Form invertiert werden. Eine Möglichkeit, dennoch Zufallszahlen zu generieren, die nach diesen Verteilungsfunktionen verteilt sind, kann in der Annäherung der inversen Verteilungsfunktion bestehen. Die Inverse der Standardnormal-Verteilung kann beispielsweise wie folgt angenähert werden:

$$F^{-1}(u) \approx \frac{u^{0,135} - (1-u)^{0,135}}{0,1975}$$

Diese Annäherung ist im Bereich zwischen 0,0012499 und 0,9986501 mindestens auf eine Dezimalstelle genau. Aus standardnormalverteilten Zufallszahlen lassen sich normalverteilte Zufallszahlen nach der Vorschrift

$$\text{ran}_{\text{Normal}} = \text{ran}_{\text{stnormal}} \cdot \sigma + \mu$$

erzeugen, wobei σ und μ die Standardabweichung beziehungsweise der Mittelwert der gewünschten Normal-Verteilung sind.

Genauere standardnormalverteilte Zufallszahlen lassen sich mit dem Box-Müller-Verfahren und dem Polarverfahren erzeugen.

6.3 Box-Müller Verfahren und Polarverfahren

Das Verfahren von Box-Müller [Devroye 1996] kann angewandt werden, um in konstant vielen Schritten normalverteilte Ereignisse zu erzeugen. Mittels zweier uniform verteilten Zufallszahlen u_1, u_2 aus dem Bereich $[0..1]$ können mittels der Bildungsvorschrift

$$x = \sqrt{-2 \ln u_1} \sin(2\pi u_2) \text{ und } y = \sqrt{-2 \ln u_1} \cos(2\pi u_2)$$

zwei standardnormalverteilte Zufallszahlen erzeugt werden. Die gewünschten normalverteilten Zufallszahlen ergeben sich dann zu

$$\text{ran}_1 = x \cdot \sigma + \mu \text{ bzw. } \text{ran}_2 = y \cdot \sigma + \mu$$

Wegen der Verwendung von Sinus und Cosinusfunktionen ist die Box-Müller Methode nicht sonderlich schnell. Eine oftmals schnellere Methode besteht in der Polarmethode [Devroye 1996]. Dazu werden mittels der uniform verteilten Zahlen u_1, u_2 die Terme

$$v_1 = 2u_1 - 1, v_2 = 2u_2 - 1 \text{ und } r = \sqrt{v_1^2 + v_2^2}$$

berechnet. Falls $r > 1$ ist, muss die Berechnung mit neuen u_1, u_2 wiederholt werden, andernfalls ergeben sich zwei standardnormalverteilte Zufallszahlen zu

$$x = v_1 \sqrt{\frac{-2 \ln r}{r}} \text{ und } y = v_2 \sqrt{\frac{-2 \ln r}{r}},$$

die wie oben zu normalverteilten Zufallszahlen transformiert werden können. Nachteil der Polarmethode ist, dass eine Berechnung nicht in konstant vielen Schritten möglich ist, da gegebenenfalls die Generierung zweier Zufallszahlen mehrmals wiederholt werden muss, bis sie der Forderung

$$1 \leq v_1^2 + v_2^2$$

genügen.

Da bei beiden Methoden jeweils zwei unabhängige [Brent 1993] Zufallszahlen generiert werden ist es zweckmäßig, bei Implementierungen, die beim Aufruf eine einzelne Zufallszahl berechnen, die jeweils nicht verwendete Zufallszahl zwischenspeichern und bei der nächsten Anfrage zurückzuliefern.

6.4 Generierung von lognormalverteilten Ereignissen

Die Lognormal-Verteilung ist definiert als:

$$Y = \log X \sim \text{Normal} \Leftrightarrow X \sim \text{Lognormal}$$

Zur Generierung von lognormalverteilten Zufallszahlen macht man sich daher die Umkehrung zu Nutze:

$$X \sim \text{Normal}(\mu, \sigma^2) \Leftrightarrow Y = e^X \sim \text{Lognormal}(\mu, \sigma^2) \text{ [Paxson 1994]}$$

Mittels einer vorhandenen Methode zur Generierung normalverteilter Zahlen wie der bereits vorgestellten Polarmethode, lassen sich mittels der Funktion

$$X = e^{\text{Normal}(U)}$$

lognormalverteilte Zufallszahlen berechnen. Analog können Zufallszahlen nach der in [Paxson 1994] verwendeten Log2normal-Verteilung mit der Funktion

$$X = 2^{\text{Normal}(U)}$$

erzeugt werden. Entsprechend wird bei der Loggumbel- und Log2gumbel-Verteilungen vorgefahren.

6.5 acceptance-rejection Methode (Poisson-Verteilung)

Zur Berechnung poissonverteilter Zufallszahlen macht man sich zu Nutze, dass N bei einer Poisson-Verteilung der Form

$$P(N = x) = \frac{e^{-\lambda} \lambda^x}{x!}$$

als Anzahl der Ereignisse eines Poisson-Ankunftsprozesses pro Zeiteinheit verstanden werden kann [Fadiloglu 2002]. Die Zeit zwischen den Ereignissen ist dann exponentiell mit der Rate λ verteilt. Aus der Ungleichung

$$N = x \Leftrightarrow \sum_{i=1}^x A_i \leq 1 \leq \sum_{i=1}^{x+1} A_i$$

mit den Werten A_i für die exponentiell verteilten Ereignisse ergibt sich für die Erzeugung von poissonverteilten Zufallszahlen die Vorschrift: Zu finden ist die Anzahl n , für die die Summe der exponentiell verteilten Werte A_i die Zahl 1 letztmals nicht überschreitet.

Eine poissonverteilte Zufallszahl kann damit mittels folgender Vorschrift erzeugt werden:

1. $x=0; s=0$
2. y ist eine $\exp(\lambda)$ verteilte Zufallszahl, wobei λ der Parameter der Poissonverteilung ist
3. $s+=y$
4. if $s>=1$ then stop, „ x ist die gewünschte Zahl“
5. else $x++$; goto 2

Nach einem ähnlichen Ansatz lassen sich auch Zufallszahlen mit geometrischer Verteilung erzeugen. Dazu werden solange uniform verteilte Zufallszahlen erzeugt, bis eine Zufallszahl mit einem Wert kleiner oder gleich dem Parameter p der Verteilung gefunden wird. Die Anzahl der benötigten Versuche ist die gewünschte Zufallszahl mit geometrischer Verteilung.

6.6 Generierung von zipfverteilten Zufallszahlen

Da die Generierung von Zipf-verteilten Zufallszahlen lediglich zur Auswahl eines der angegebenen HTTP-Server benötigt wird, genügt es die (langsame) Methode der inversion-rejection Methode zu implementieren, bei der numerisch die inverse Verteilungsdichtefunktion gebildet wird. Dazu wird das Ereignis gesucht, ab dem die inverse Verteilungsdichtefunktion eine zuvor gewählten Wert mit uniformer Verteilung überschreitet. Dies ist möglich, da es sich bei der Zipf-Verteilung um eine diskrete Verteilung handelt, was die numerische Berechnung der Verteilungsdichtefunktion sehr vereinfacht.

6.7 Inverse Transformation bei heuristischen Verteilungen

Eine Möglichkeit zur Erzeugung von Zufallszahlen, die nach einer vorgegebenen heuristischen Verteilung verteilt sind, wird in [Danzig, Jamin 1991] genannt. Bei den von tcplib verwendeten heuristischen Modellen, die nicht in einer geschlossenen analytischen Formel angegeben werden, ist eine direkte Berechnung der inversen Verteilungsfunktion verständlicherweise nicht möglich. Tcplib erstellt zur Berechnung mittels der inversen Transformation daher eine abschnittsweise lineare Annäherung an die Verteilungsfunktion. Dazu wird aus den gemessenen Daten des heuristischen Modells mittels eines Histogramms eine Verteilungsfunktion berechnet und in einem Feld abgelegt. Da die Verteilungsfunktion, wie bereits erwähnt, monoton wächst, ist das Feld bereits sortiert. Damit kann ein Wert *ran*, der

zu einer uniform verteilten Zufallszahl u berechnet werden soll, nun durch einen beliebigen Suchalgorithmus (beispielsweise binäre Suche) über das Verteilungsarray ermittelt werden.

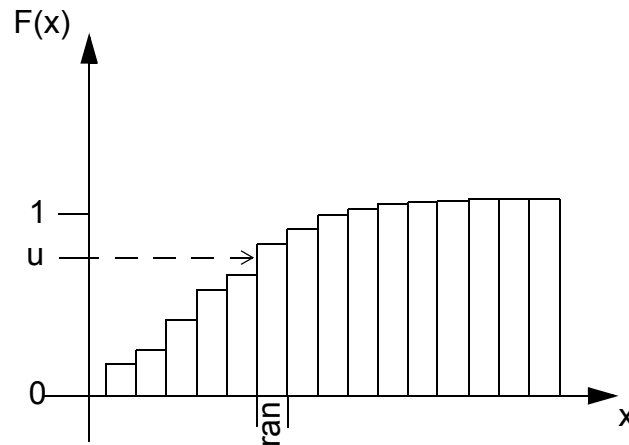


Abb. 22 Inverse Transformation bei heuristischen Modellen

Als Verbesserung dieses Verfahrens wird die Variation genannt, anstatt des Dichtearrays ein Quantilenarray (also eine lineare Annäherung an die inverse Dichtefunktion) mit vorgegebener Schrittweite zu verwenden. Dies hat den Vorteil eines konstanten (und wählbaren) Wertebereichs. Im Gegensatz zum Verfahren mit Dichtefeld, bei dem das Feld so viele Elemente enthält, wie im heuristischen Modell unterschiedliche Werte vorkommen, ist die Anzahl der Elemente beim Quantilenfeld durch gewünschte Genauigkeit wählbar. Zum Beispiel können 10 Elemente mit einer Erhöhung der Quantile um jeweils 0.1 verwendet werden. Bei fester Schrittweite ist die Berechnung eines Wertes *ran* mittels eines gleichverteilten Wertes u und des Quantilenfelds in konstanter Zeit möglich, da kein Suchalgorithmus benötigt wird.

7 Diskussion der Anwendungsmodelle

Bei allen implementierten Modellen außer dem HTTP-Modul wurde auf eine Modellierung der Sitzungsdauer und der Serverwahl verzichtet. Die Sitzungsdauer wird durch den Start und das Beenden der Lasterzeugung kontrolliert. Dies ist notwendig, da ansonsten nur beim Betrieb einer sehr großen Anzahl von Lasterzeugungsmodulen eine dauerhafte Last vorhanden wäre. Die Pausen zwischen zwei aufeinander folgenden Sitzungen eines Clients sind oftmals im Stundenbereich. Da mit sehr langen Pausen eine sinnvolle Lasterzeugung nur bei sehr vielen beteiligten Knoten möglich wäre, in der Zielplattform, dem NET-Cluster, jedoch lediglich 64 Knoten geplant sind, wurde auf die Modellierung der Sitzungsebene verzichtet.

7.1 FTP

Um eine Überlastung eines Knoten durch die vom FTP-Protokoll nicht begrenzte Datenrate auszuschließen, wird serverseitig die Datenrate begrenzt. Dies ist notwendig, da ein FTP-Client wegen seiner bulktransfer-Eigenschaften die zur Verfügung stehende Bandbreite voll ausnutzt, und da Java bei hoher IO-Last die CPU zu 100 Prozent belastet. Dies verfälscht nicht notwendigerweise die Verkehrscharakteristika, da oftmals auch (vor allem öffentlich zugängliche) FTP-Server die Bandbreite pro Clientverbindung beschränken, um einer Überlastung durch einzelne, gut angebundene, Client-Systeme vorzubeugen.

Das vorliegende FTP-Lastmodell beruht auf Messungen aus den Jahren 1990 bis 1994 und dürfte damit nur noch bedingt auf heutige Anwendungsfälle anwendbar sein. Dies ist vor allem darauf zurückzuführen, dass sich das Nutzungsverhalten bei FTP stark geändert hat. Während 1994 das FTP-Protokoll das Standardprotokoll zur Dateiübertragung war - in [Paxson 1994] hatte FTP einen Anteil von 50% bis 80% am Umfang der übertragenen Daten - haben heute das HTTP-Protokoll und die Protokolle verteilter Dateisysteme wie SMB und NFS diese Bedeutung übernommen. Zudem ist die durchschnittliche Dateigröße seit 1994 stark angestiegen, da große Datenträger sowie breitbandige Rechnernetze eine strenge Limitierung der Dateigröße unnötig machen.

FTP wird heute vor allem zur Übertragung großer Datenmengen wie CD-Images verwendet oder zur Aktualisierung eines Webservers. Während bei der zweiten Anwendungsart das Lastmodell vergleichbar zu den Daten von 1994 ist, so ist es bei der ersten Anwendung völlig verändert. Insbesondere fehlt oft die Navigation im Verzeichnisbaum. Da heute oftmals eine FTP-Datei direkt durch einen Link eines Webservers angesprochen wird, ist das Navigieren im Verzeichnisbaum bei vielen FTP-Sitzungen ganz weggefallen.

Mit Hilfe von neuen Messungen wird es jedoch vermutlich möglich sein, durch Ermittlung der momentan gültigen Verteilungsparameter für Datei- und Sitzungsgröße den Rest des Last-Modells anzupassen.

7.2 HTTP

Im Lasterzeugungsmodul wird das Konzept der persistenten Verbindungen (HTTP 1.1) implementiert. Das bedeutet, dass eine TCP-Verbindung nicht wie in der ursprünglichen Version des HTTP 1.0 Protokolls nach Übertragung eines Objekts geschlossen wird, sondern für weitere Übertragungen zur Verfügung steht. Weiterhin werden mehrere TCP-Verbindungen für ein Client-Server Paar unterstützt, um die gleichzeitige Übertragung

mehrerer Objekte zuzulassen. Die Anzahl der maximal möglichen Verbindungen pro Client ist konfigurierbar.

Wie bereits bei FTP, wird auch bei HTTP serverseitig eine Lastbegrenzung durch Angabe einer maximalen Bitrate pro TCP-Verbindung implementiert. Auch bei HTTP-Servern ist dies nicht unüblich, um einer Überlastung des Servers durch einzelne Clients vorzubeugen [mod_throttle].

Da die Parameter der charakteristischen Verteilungen aus dem Jahr 1998 stammen, dürften sie inzwischen nur noch begrenzt Gültigkeit besitzen. Vor allem die Anzahl der Objekte pro Dokument und die Dateigröße der sekundären Verbindungen dürften heute aufgrund stark grafiklastiger WWW-Dokumente gestiegen sein. Es ist jedoch kein Grund zu erkennen, warum das grundsätzliche Modell keine Gültigkeit mehr besitzen sollte, somit müsste es möglich sein, das Modell an heutige Verhältnisse durch Messung der aktuell gültigen Parameter anzupassen.

Im HTTP-Lastgenerator werden Objekte, die von einem anderen Servern als dem aktuell ausgewählten eingebettet werden, nicht nachgebildet. Um dies zu unterstützen müsste eine serverübergreifende Verteilung der Beliebtheit von Dokumenten und Objekten berechnet werden, anstatt, wie im vorliegenden Fall, lediglich eine Beliebtheit der Server zu modellieren. Ein Ansatz mit der Modellierung der Beliebtheit von Dateien wird in [Barford, Crovella 1997] verfolgt.

7.3 Telnet

Bei den Mitschnitten, die Grundlage für die Erstellung der heuristischen Modelle von tcplib - und damit auch der in dieser Arbeit verwendeten - waren, wurde der Inhalt der Datenpakete nicht ausgewertet. Lediglich die TCP und IP Header konnten zur Erstellung der heuristischen Modelle verwendet werden. Daher unterscheidet das Modell zur Berechnung der Paketankunftsrate nicht zwischen Paketen, die während der Eingabe eines Befehls übertragen werden (also die eingegebene Zeichen), und der Wartezeit zwischen zwei Befehlen. Daher fließen die langen Wartezeiten zwischen zwei Befehlen in das Paketankunfts-Modell mit ein, so dass während der Eingabe von Befehlen (subjektiv) unnatürlich lange Pausen entstehen. Dieses Problem lässt sich mit dem vorliegenden Modell (insbesondere mit den vorliegenden Messwerten aus dem tcplib-Projekt) nicht beheben. Auf den modellierten Datenverkehr hat dies jedoch nur geringe Auswirkungen, lediglich die Pausen nach einem Befehl (nach Erhalt der Antwort des Servers) dürften zu gering sein.

7.4 MPEG Videostrom

Die bei der Erstellung des MPEG-Videomodells verwendeten Daten stammen aus dem Jahr 1995. Da die Videokompressionstechnik seither deutlich verbessert wurde, dürften die Verteilungsparameter der Rahmengrößen (die sich aber einfach anpassen lassen) heute nicht mehr gültig sein. Insbesondere gilt dies für Videoformate, die für die Verwendung des Filmes auf Geräten mit schmalbandiger Anbindung optimiert sind. Dazu zählen MPEG4, das unter anderem für die Wiedergabe auf Handheld-Computern oder Mobiltelefonen ausgelegt ist, oder das DivX-Format, das vor allem bei Filmen in VHS-ähnlicher Qualität verbreitet ist. Für Filme in DVD-Qualität sind die Verteilungsparameter der Rahmengrößen aus [Krunz et al. 1996] weiterhin gültig.

Schwerer wiegt jedoch das Fehlen variabler GOP-Sequenzen. Videokodierer sind heute in der Lage, die GOP-Sequenzen an die wechselnde Dynamik des zu komprimierenden Films

anzupassen, um damit einerseits eine höhere Kompression des Ausgangsmaterials bei Szenen mit geringer Dynamik zu erreichen, andererseits die Qualität bei Szenen mit hoher Dynamik zu verbessern, indem die GOB-Sequenzen verkürzt werden. Eine Möglichkeit, variable GOB-Sequenzen zu implementieren wird in [Agrawal et al. 1997] genannt.

7.5 Onlinespiele

Das verwendete Lastmodell berücksichtigt nicht die Korrelation zwischen eingehenden und ausgehenden Paketen und modelliert damit auch nicht die Verarbeitungszeit des Servers zwischen Eingang der Meldungen der Clients und Übertragung der berechneten Änderungen der Spielwelt zu den Clients. Diese dürfte jedoch bei den unterschiedlichen Spielen stark von den einzelnen Kommunikationsbibliotheken abhängig sein, so dass ohne weitergehendes Wissen über die (im allgemeinen weder standardisierten noch offengelegten) Kommunikationsprotokolle der Spiele kein besseres Verhalten des Lastgenerators erzeugt werden kann.

Möglicherweise ist es auch sinnvoll, den Einfluss der nicht-interaktiven Phase der Kommunikation zur untersuchen, also den Versionsabgleich der Spielwelt vor Beginn des Spiels. Teilweise werden bei diesem Abgleich nicht unerhebliche Datenmengen übertragen. Insbesondere bei Counterstrike haben die Spieler die Möglichkeit, neue Spielwelten zu erstellen, die dann vor Spielbeginn an alle teilnehmenden Spieler verteilt werden.

8 Zusammenfassung und Ausblick

Diese Arbeit behandelte die Erstellung eines Lastgenerators zum Einsatz in der Netzemulationsumgebung „Network Emulation Testbed“. Es wurden verschiedene Ansätze zur Beschreibung und Generierung von Netzlast beschrieben. Weiterhin wurden Modelle mehrerer Lasttypen beschrieben und innerhalb eines Lastgenerators implementiert.

Die Modelle, die für die Beschreibung der Anwendungslasten und zur Generierung der synthetischen Netzlast verwendet wurden, sind teilweise über fünf Jahre alt. Da aufgrund eines veränderten Benutzerverhaltens und der immer noch stark anwachsenden verfügbaren Bandbreite der Netzwerkverkehr stark gestiegen ist, müssen die Modelle verifiziert und gegebenenfalls mit Hilfe neuer Messdaten aktualisiert werden. Es ist jedoch wahrscheinlich, dass der grundsätzliche Aufbau der Modelle übernommen werden kann, und nur die Verteilungen der charakteristischen Größen angepasst werden müssen.

Die Gewinnung neuer Modelle ist eine nicht-triviale Aufgabe. Die üblicherweise angewandte Methode des Mitschneidens von Netzwerkverkehr muss möglichst an einem zentralen Punkt eines Netzes durchgeführt werden, um nicht verfälschte Messdaten durch abteilungsspezifische Besonderheiten zu erhalten. Durch die stark gestiegene Bandbreite in den letzten Jahren wird dies jedoch zunehmend schwierig.

Während Moores Gesetz von der Verdopplung der Leistungsfähigkeit von CPUs innerhalb von 18 Monate weiterhin Gültigkeit besitzt, wird beim Wachstum des Datenverkehrs im Internet von einer Verdopplung innerhalb zwölf Monaten ausgegangen [Coffman, Odlyzko 2001]. Andere Arbeiten nennen noch höhere Wachstumsraten.

In Routern mit breitbandiger Anbindung müssen daher hochspezialisierte ASICs (Application Specific Integrated Circuit) eingesetzt werden, um das stärkere Wachstum der Bandbreite im Vergleich zu dem der CPU-Leistung zu kompensieren. Es ist daher heute nicht mehr ohne Weiteres möglich, den gesamten Datenverkehr einer großen Gruppe von Anwendern zu analysieren.

Daher wird es gegebenenfalls notwendig sein, mehrere Messungen bei unterschiedlichen Anwendergruppen durchzuführen, und aus den Teilergebnissen ein neues Modell zu entwickeln. Möglicherweise kann es auch zweckmäßig sein, die Netzwerkprotokollierung durch Messungen am Server oder Client zu ersetzen oder zu ergänzen.

9 Literaturangaben

[Adas 1997]

Abdelnaser Adas, „**Traffic Models in Broadband Networks**“, IEEE Communications Magazine, Juli 1997, Seiten 82-89

[Agraval et al. 1997]

Sanjey K. Agrawal, Charles F. Barry, Vinay Bannai, Leonid Kazavsky, „**Characterization, modeling and multiplexing of real-time MPEG-II video**“, in: Proceedings of the 5th International Conference on Telecommunication Systems Modeling and Analysis, Nashville, 20-23 März 1997, Seiten 431ff

[Almeida et al. 1996]

Virgilio Almeida, Azer Bestavros, Mark Crovella, Adriana de Oliveira, „**Characterizing Reference Locality in the WWW**“, In: Proceedings of PDIS'96: The IEEE Conference on Parallel and Distributed Information Systems, Miami Beach, Dec 1996

[Barford, Crovella 1997]

Paul Barford, Mark Crovella, „**An Architecture for an WWW Workload Generator**“, World Wide Web Consortium Workshop on Workload Characterization, October 1997

[Barford, Crovella 1998]

Paul Barford, Mark Crovella, „**Generating Representative Web Workloads for Network and Server Performance Evaluation**“, In: Proceedings of the ACM International Conference in Measurement and Modeling of Computer Systems, Madison, Seiten 151-160

[Borella 1999]

Michael S. Borella, „**Source Models of Network Game Traffic**“, Proceedings, Network+Interop '99 Engineer's Conference, May 1999

[Braganza, Pamidi 2000]

Fernand Braganza, Praveen Pamidi, „**Quake 2 Network Analysis and Performance**“, Projektbericht, University of California, San Diego, 22.03.2000

[Brent 1993]

Richard P. Brent, „**Fast Normal Random Number Generators for Vector Processors**“, Australian National University Canberra, Computer Sciences Laboratory, Technical Report TR-CS-93-04, März 1993

[Catledge 1995]

Lara D. Catledge, James E. Pitkow, „**Characterizing browsing strategies in the World-Wide Web.**“, In: Proceedings of the Third International World Wide Web Conference, Darmstadt, April 1995.

[Charzinski et al. 2002]

J. Charzinski, J. Färber, N. Vicari, „**Verkehrsmessungen und Lastmodellierung im Internet**“, Praxis der Informationsverarbeitung und Kommunikation (PIK), Vol. 25.2, April-Jun 2002, S. 64-72

[Coffman, Odlyzko 2001]

K. G. Coffman, A. M. Odlyzko, „**Growth of the Internet**“, AT&T Labs Research

[Danzig, Jamin 1991]

Peter B. Danzig, Sugih Jamin, „**tcplib: A Library of TCP Internetwork Traffic Characteristics**“, USC Networking and Distributed Systems Laboratory TR CS-SYS-91-01, Oktober 1991

[Danzig et al. 1992]

Danzig, S. Jamin, R. Caceres, D. Mitzel, and D. Mestrin, "**An Empirical Workload Model for Driving Wide-Area TCP/IP Network Simulations**", Internetworking: Research and Experience, vol. 3, no. 1, pp. 1--26, 1992.

[Devroye 1996]

Luc Devroye, „**Random Variate Generation in one line of code**“, In Proceedings of the 1996 Winter Simulation Conference Proceedings, 1996, Seiten 265-272

[Fadiloglu 2002]

Mehmet Murat Fadiloglu, „**IE 324 -- Simulation**“, Vorlesungsunterlagen Simulation Frühjahr 2002, Industrial Engineering Dpt, Bilkent Universtät, Ankara

[Faerber 2002]

Johannes Färber, „**Network Game Traffic Modelling**“, Netgames 2002, Braunschweig 16-17. April 2002

[Heegaard 2000]

Poul E. Heegaard, "**GenSyn - a Java-based Generator of Synthetic Internet Traffic Linking User Behavior Models to Real Network Protocols**", Presentation at ITC Specialist Seminar on IP Traffic Measurement, Modeling and Management, Monterey, 18-20 September 2000

[Herrscher et al.2002]

Daniel Herrscher, Alexander Leonardi, Kurt Rothermel, „**Modeling Computer Networks for Emulation**“, In: Proceedings of the 2002 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'02), Las Vegas, Juni 2002, Seiten 1725-1731,

[Herrscher, Rothermel 2002]

Daniel Herrscher, Kurt Rothermel, „**A Dynamic Network Scenario Emulation Tool**“, In: Proceedings of the 11th International Conference on Computer Communications and Networks (ICCCN 2002), Miami, Oktober 2002, Seiten 262-267

[Ishac 2001]

Joseph Ishac, „**FTP Traffic Generator**“, Technical Report, Department of Electrical Engineering and Computer Science, Case Western Reserve University, Januar 2001

[Jain, Routhier 1986]

R. Jain and S. Routhier, "**Packet Trains-Measurements and a New Model for Computer Network Traffic**", IEEE Journal of Selected Areas in Communications, Vol. SAC-4, No. 6, September 1986

[JMeter]

Apache Software Foundation, The Jakarta Project, „**JMeter, application to load test functional behavior and measure performance**“, <http://jakarta.apache.org/jmeter/index.html>

[Krunz et al. 1995]

Marwan Krunz, Ron Sass, Herman Hughes, „**Statistical Characteristics and Multiplexing of MPEG Streams**“, In: Proceedings of the IEEE INFOCOM '95 Conference, Boston, Apr. 1995, Seiten 455-462

[Krunz et al. 1996]

Marwan Krunz, Satish K. Tripathi, „**Scene-Based Characterization of VBR MPEG-Compressed Video Traffic**“, University of Maryland, Department of Computer Science Technical Report CS-TR-3573, 1996

[Lucas et al. 1997]

Matthew Lucas, Bert Dempsey, Dallas Wrege, Alfred Weaver, **“An Efficient Self-Similar Traffic Model for Wide-Area Network Simulation”**, IEEE GLOBECOM '97, Phoenix, AZ, November 1997. TECHNICAL RESEARCH REPORT

[Mah 1997]

Bruce A. Mah, **„An Empirical Model of HTTP Network Traffic“**, In; Proceedings of the IEEE InfoCom 1997, Kobe, April 1997, Seiten 592-600

[mod_throttle]

Anthony Howe, **„Bandwidth & Request Throttling for Apache 1.3“**, http://www.snert.com/Software/mod_throttle/index.shtml

[Ramakrishnan 1999]

Pradeep Ramakrishnan, **„Self-Similar Traffic Models“**, CSHCN, University of Maryland

[Paxson 1994]

Vern Paxson, **„Empirically Derived Analytic Models of Wide-Area TCP Connections“**, IEEE/ACM Transactions on Networking, Vol. 2 No. 4, August 1994, Seite 316 ff

[Paxson 1994-2]

Vern Paxson, **„Growth Trends in Wide-Area TCP-Connections“**, IEEE Networks, 8(4), Juli/August 1994, S. 8-17

[Paxson 1995]

Vern Paxson, Sally Floyd, **„Wide-Area Traffic: The Failure of the Poisson Modeling“**, IEEE/ACM Transactions on Networking, 3(3), Seiten 226-244, Juni 1995

[Paxson 1996]

Vern Paxson, **„An introduction to Internet Measurement and Modeling“**, ACM SIGCOMM 1996 Tutorial

[Steinmetz 1996]

R. Steinmetz, **„Human perception of jitter and media synchronization“**, IEEE Journal on Selected Areas in Communications, 14(1), Januar 1996, Seiten 61-72

[Tanenbaum 1996]

Andrew S. Tanenbaum, **„Computer Networks, 3rd Edition“**, New Jersey: Prentice Hall, Inc, International Editions, 1996

[Taqqu et al. 1997]

Murad S. Taqqu, Walter Willinger, Robert Sherman, **„Proof of a Fundamental Result in Self-Similar Traffic Modeling“**, ACM/SIGCOMM Computer Communication Review, Vol. 27, April 1997, Seiten 5-23

[tcpdump]

Lawrence Berkeley National Laboratory, University of California, Berkeley, **“tcpdump - dump traffic on a network”**, <http://www.tcpdump.org>

Ich versichere, dass ich diese Arbeit selbständig verfasst und nur die angegebenen Hilfsmittel verwendet habe.

<Unterschrift>