Messung und Modellierung der von WWW in Zugangsnetzen erzeugten Verkehrslast

Tim Irnich

20. Juli 2000

1. Einleitung
2. Verhalten von Browsern und Benutzern
3. Messergebnisse
4. Modellierung
5. Zusammenfassung
6. Literatur

Motivation

- **Anteil von WWW am gesamten Internetverkehr hat drastisch zugenommen** (in aktuellen Arbeiten werden Werte zwischen 75% und 96% genannt).
  ⇒ Der zusätzliche Aufwand durch genaue Modellierung des WWW-Verkehrs lohnt sich, da die Charakteristiken von WWW den Gesamtverkehr entscheidend beeinflussen.

- **Grundproblematik bei der Modellierung:**
  In einer bestimmten Umgebung gemachte Beobachtungen lassen sich nur eingeschränkt verallgemeinern.
  ⇒ **Suche nach Gemeinsamkeiten wird zur Hauptaufgabe**

Vorüberlegungen

- **Kleinster gemeinsamer Nenner: HTTP und TCP**
  $\Rightarrow$ Genaue Modellierung des Zusammenspiels dieser beiden Protokolle bietet hinreichend allgemeine Grundlage.
- **Ziel: umgebungsspezifische Anpassungen ausschließlich durch Modell-Parameter.**
- **Ein bekanntes Modell basiert auf Mosaic** [1].
  $\rightarrow$ Einfluß neuer Technologien (Java, Frames, etc.) wird nicht abgebildet. Die Architektur des Modells sollte es ermöglichen, diese Effekte ausschließlich statistisch zu erfassen.
- Eine grobe Klassifizierung von Umgebungen, die ein bestimmtes Benutzerverhalten hervorrufen, wird benötigt, um bei der Simulation eines solchen Szenarios auch ein hierfür typisches Lastaufkommen generieren zu können.

Ziele bei der Entwicklung neuer Modelle

- **Interaktion des Benutzers mit dem Web-Browser ist Ausgangspunkt aller Vorgänge.**
  $\Rightarrow$ Benutzeranfragen (sog. *Web Requests*) bilden grundlegende Einheit des Modells.
- Anstieg des Verkehrsvolumens durch:
  - steigende Benutzerzahlen
  - zunehmende Komplexität von HTML-Dokumenten

  $\Rightarrow$ **Anstieg des Signalisierungs-Overheads bei Auf- und Abbau von TCP-Verbindungen**
- **Über 50% aller WWW-Pakete enthalten keine Nutzdaten.**
  $\Rightarrow$ Die Steuerung des TCP-Protokolls durch HTTP sollte möglichst genau abgebildet werden.

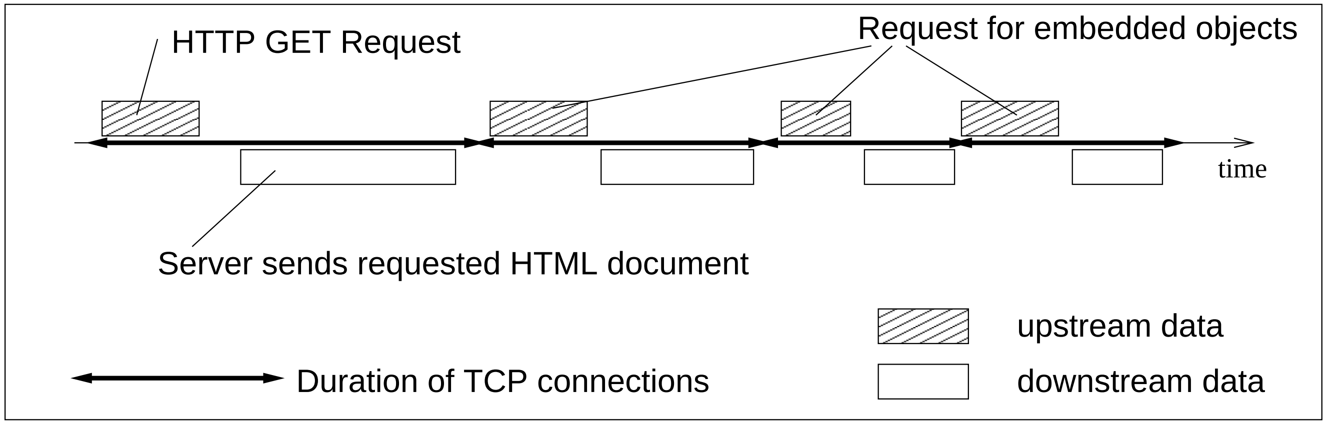
Verhalten moderner Web-Browser

- Seit HTTP/1.0 **gleichzeitiges Betreiben mehrerer TCP-Verbindungen** möglich
- **HTTP/1.0 keep-alive** bzw. **HTTP/1.1 persistent-connection**, d.h. es können **mehrere Objekte in einer TCP-Verbindung** übertragen werden.
- **Paralleler Betrieb mehrerer Browser-Instanzen** durch einen Benutzer
  $\Rightarrow$ zeitliche Verschachtelung von Web-Requests möglich.
- Verbindungen, die nach Transfer eines Objekts nicht vom Browser beendet werden, werden zumeist nach kurzer Zeit (ca. 10 sek.) serverseitig ausgelöst
- Zusätlich existiert eine Vielzahl von Einflußfaktoren auf das Benutzerverhalten, die bei der Modellierung berücksichtigt werden müssen.

Klassifizierung von Client-Server-Kommunikation

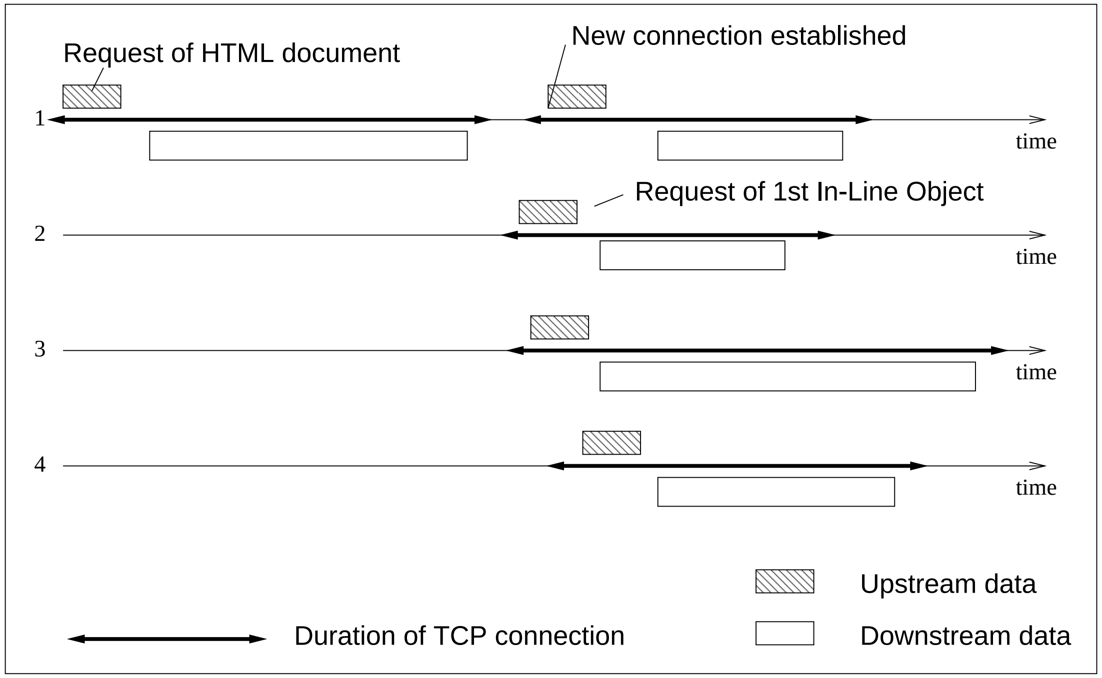
1. **HTTP/1.0, keine gleichzeitigen Verbindungen, Verbindung wird nach Transfer eines Objekts geschlossen.**
2. HTTP/1.0 mit parallelen Verbindungen, Transfer von einem Objekt pro Verbindung.
3. **HTTP/1.0 mit parallelen Verbindungen + keep-alive-Mechanismus.**
4. HTTP/1.1, parallele Verbindungen mit *persistent-connection*-Mechanismus,
   Pipelining von Web-Requests auf einer Verbindung.
   In gängigen Implementierungen: eine Pipeline für alle vorhandenen Verbindungen.
5. Bisweilen keine klare Klassifizierung möglich
   $\Rightarrow$ Modellierung durch Wahrscheinlichkeit, daß eine Verbindung nach Transfer eines Objekts geschlossen wird.

1. HTTP 1.0 „ohne alles“



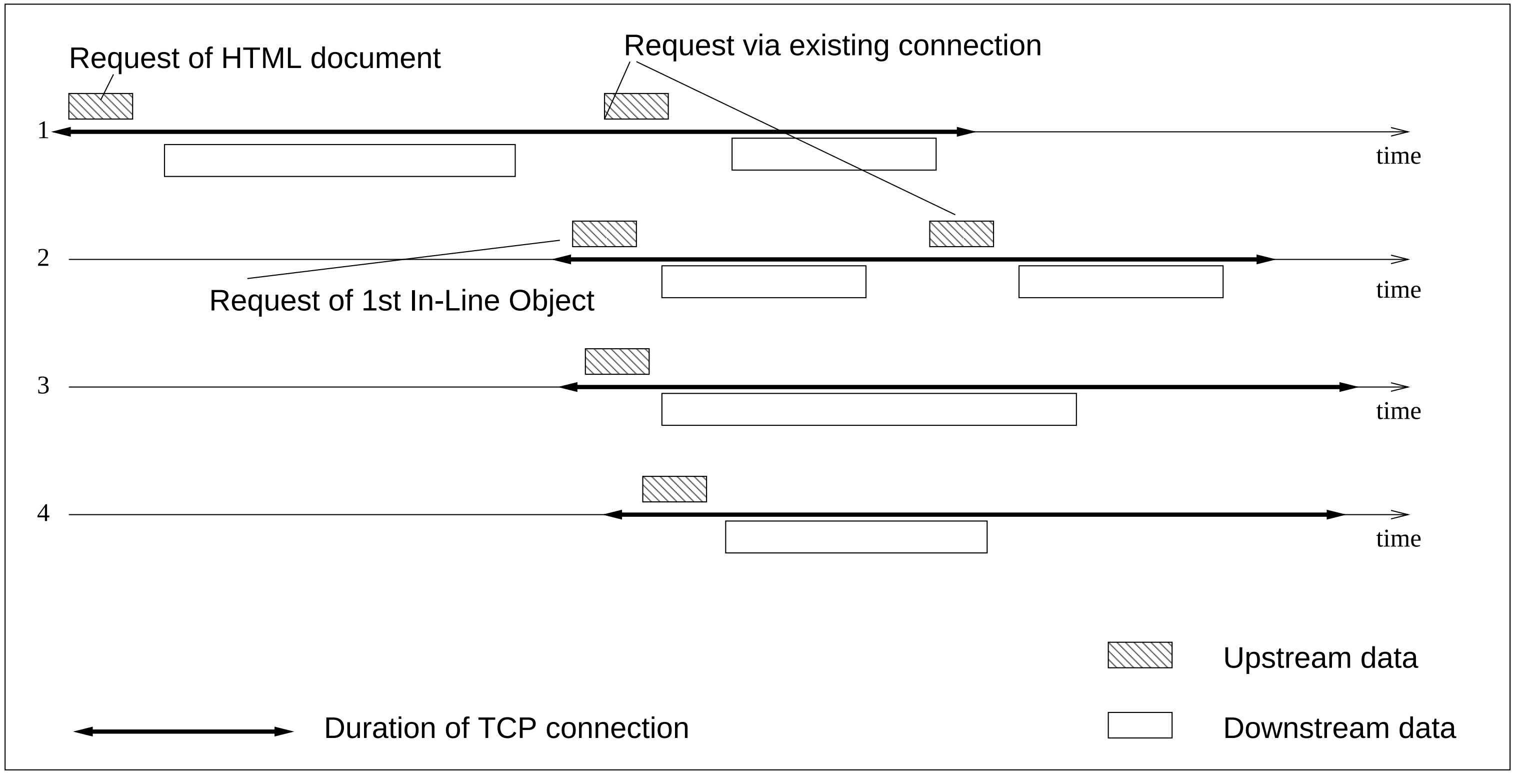
- Für jedes Objekt wird eine neue TCP-Verbindung aufgebaut.
  $\Rightarrow$ Signalisierungs-Overhead
- Jeder Client unterhält zu jedem Zeitpunkt nur maximal *eine* TCP-Verbindung
- Kurze Verbindungsdauern
- Kurze Zwischenankunftszeiten

2. HTTP 1.0 mit parallelen Verbindungen



- gleicher Signalisierungs-Overhead wie Variante 1
- Kürzere Download-Zeit
- Höherer Bandbreitebedarf
- Kurze Verbindungsdauern
- Extrem kurze Zwischenankunftszeiten

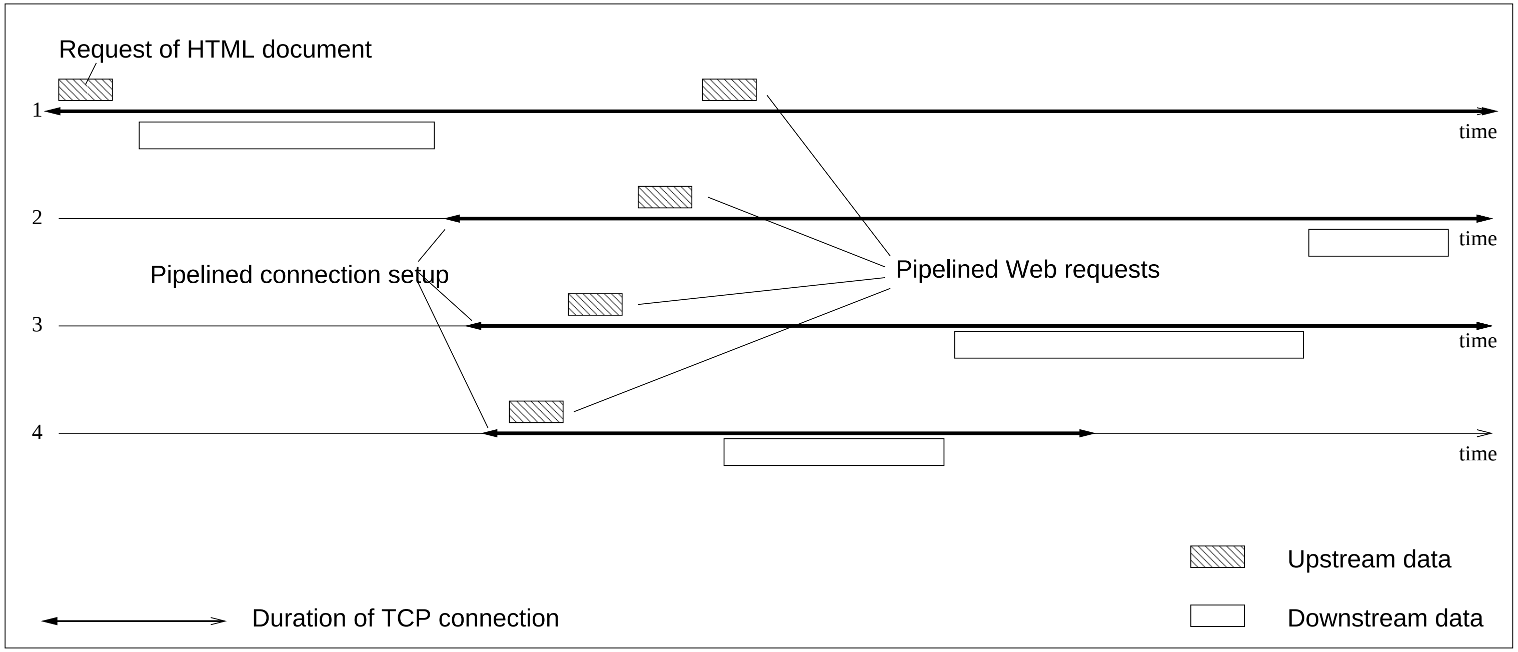
3. HTTP 1.0 mit parallelen Verbindungen + keep-alive



3. HTTP 1.0 mit parallelen Verbindungen + keep-alive (2)

- Verbindungen können nach Transfer eines Objekts bis zum Timeout leerlaufen.
- Timeout kann sowohl vom Server als auch vom Client definiert sein.
- Transfer mehrerer Objekte über eine Verbindung möglich
  $\Rightarrow$ grössere Datenmengen pro Verbindung
- Jeder der Verbindungspartner kann jederzeit die Verbindung auslösen.
- **Grosse Varianz der Zwischenankunftszeiten.**
- **Zwischenankunftszeiten zwischen erster und zweiter TCP-Verbindung sind anders verteilt als die restlichen Zwischenankunftszeiten.**

4. HTTP 1.1, persistente Verbindungen und Pipelined Requests



4. HTTP 1.1, persistente Verbindungen und Pipelined Requests (2)

- Effekt: Längere Verbindungsdauern, mehr Leerlauf
- **Benutzung der vorhandenen TCP-Verbindungen ist nicht standardisiert**
- Vorteil des Pipelining-Mechanismus: vor allem auf der Serverseite wird weniger Rechenleistung für einen einzelnen Client benötigt (Lastverteilung).

Einflüsse der Umgebung auf das Benutzerverhalten

Einflußfaktoren:

- Abrechnungsmethode:
  - zeitbasiert
  - volumenbasiert
  - Flat Rates
  - Always On/Dynamic ISDN
- Antwortzeiten („Web Performance“)
- Bandbreite des Internetzugangs
- Hardware/Software-Plattform

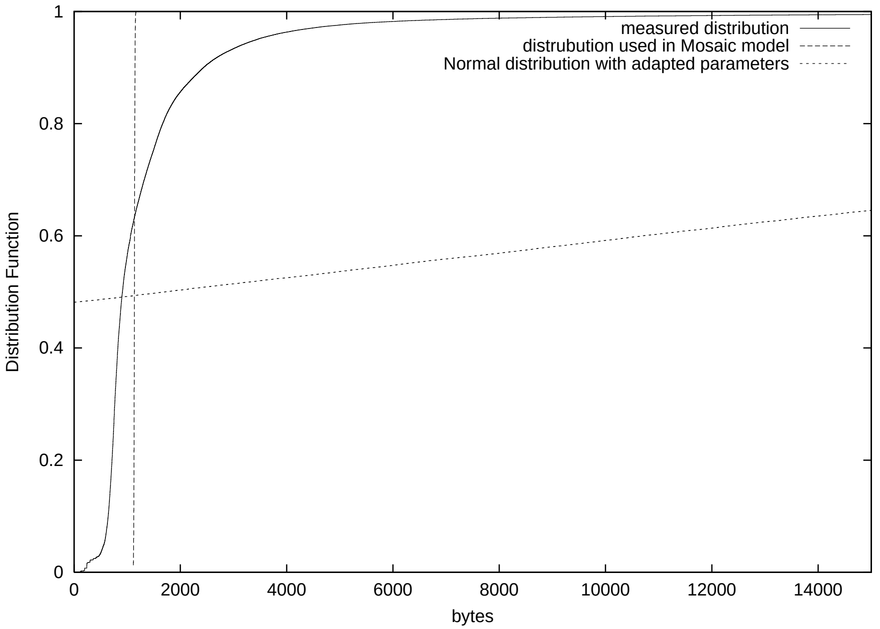
3. Messergebnisse

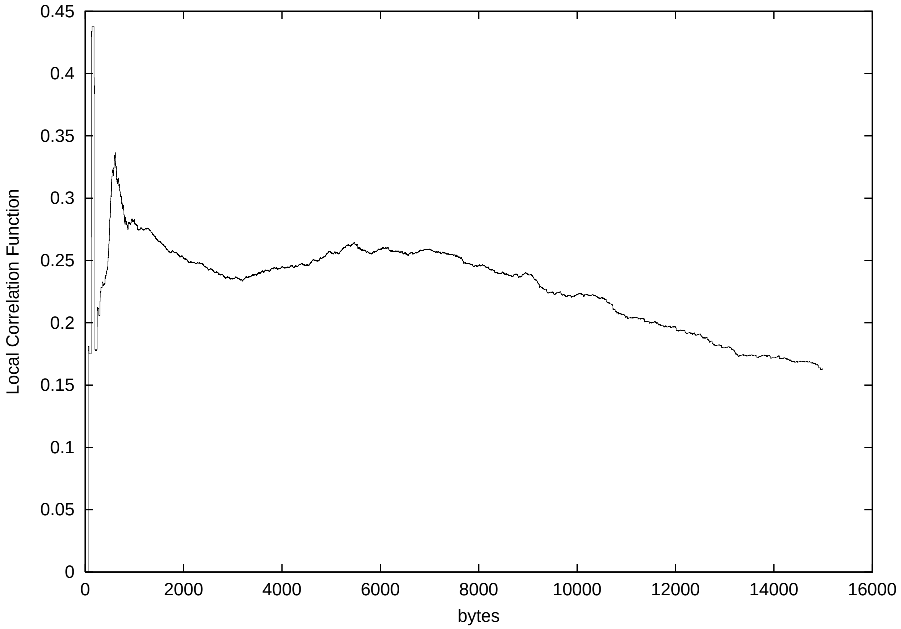
Gemessene Parameter:

- **Pro Verbindung übertragene Datenmenge in Bytes, getrennt nach upstream und downstream.**
- **Zwischenankunftszeiten der Verbindungen, die von derselben IP-Adresse ausgehen.**
- Anzahl der in einer Verbindung übertragenen Objekte.
- Paketgrößen der HTTP-GET-Requests.
- **Verzögerung des ersten GET-Requests nach Aufbau einer Verbindung.**

Die Ergebnisse wurden mit Hilfe des LRE-Algorithmus [3] gewonnen. Dadurch konnte neben der Verteilungsfunktion und wichtigen Momenten auch der sog. *Lokale Korrelationskoeffizient* gemessen werden.

Datenmenge pro Verbindung **upstream**

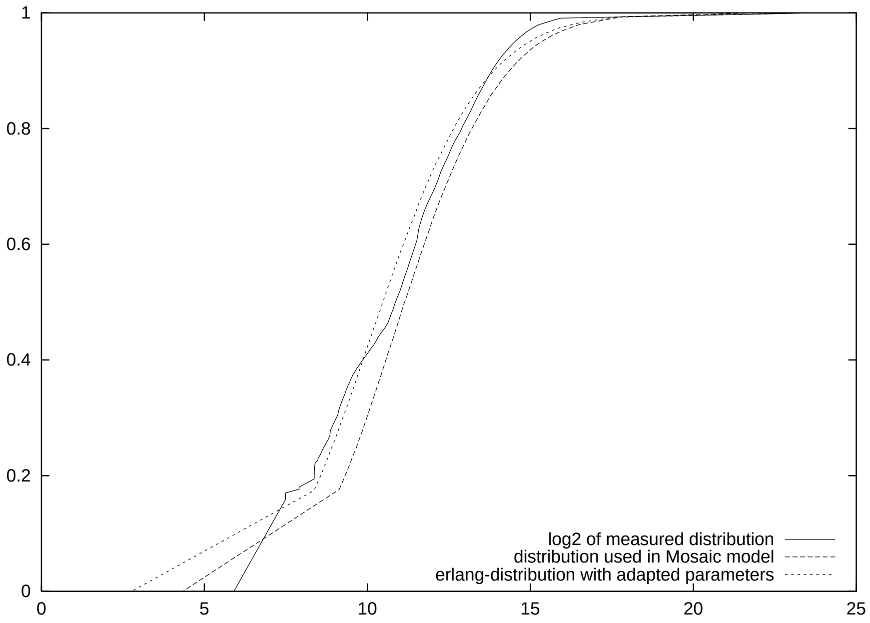


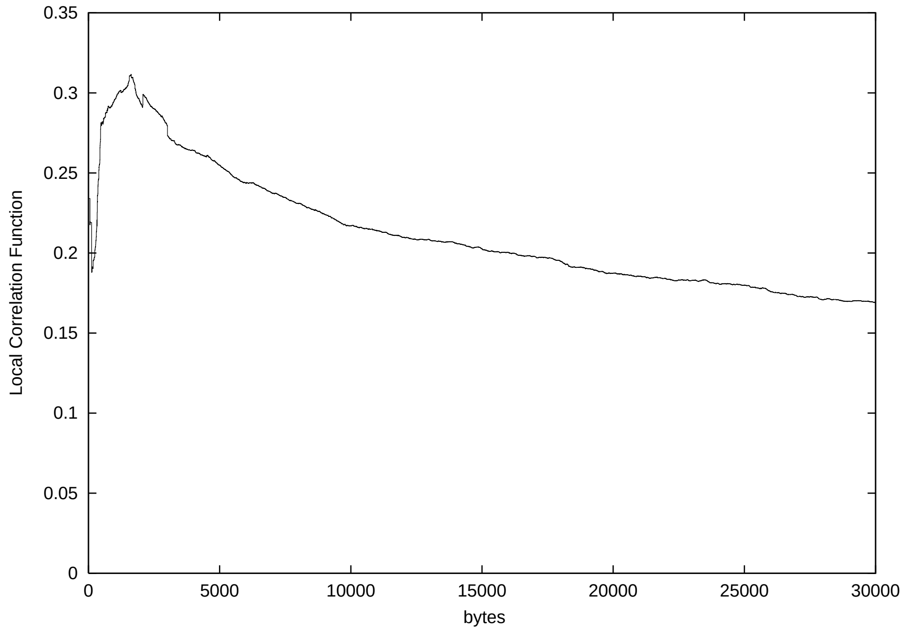


- Im Upstream vorrangig Quittungen und URL-Anfragen
- Verteilungstyp hat sich geändert

- Höherer Mittelwert, extrem große Varianz
- Besonders bei Werten < 1000 Byte starke Korrelation

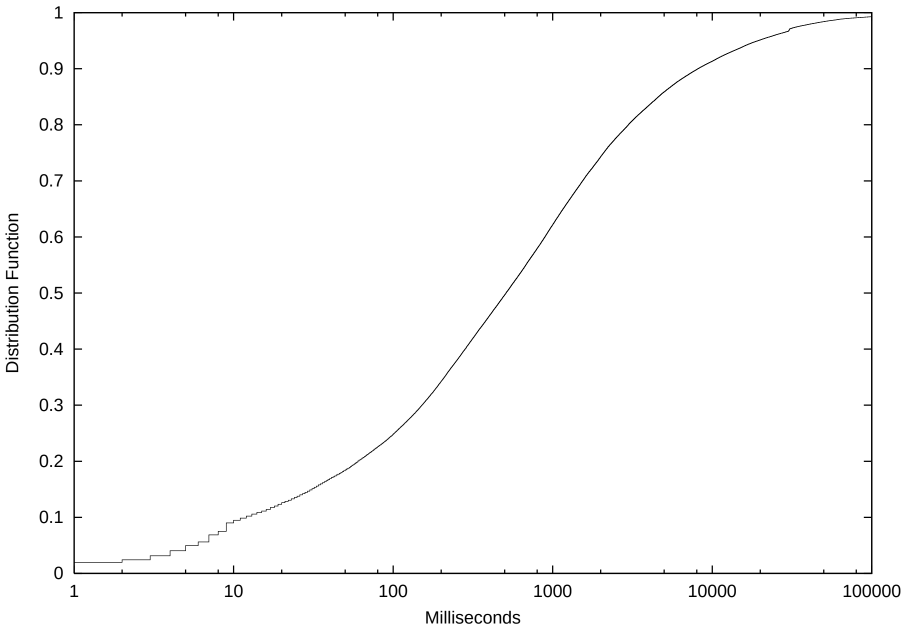
Datenmenge pro Verbindung **downstream**

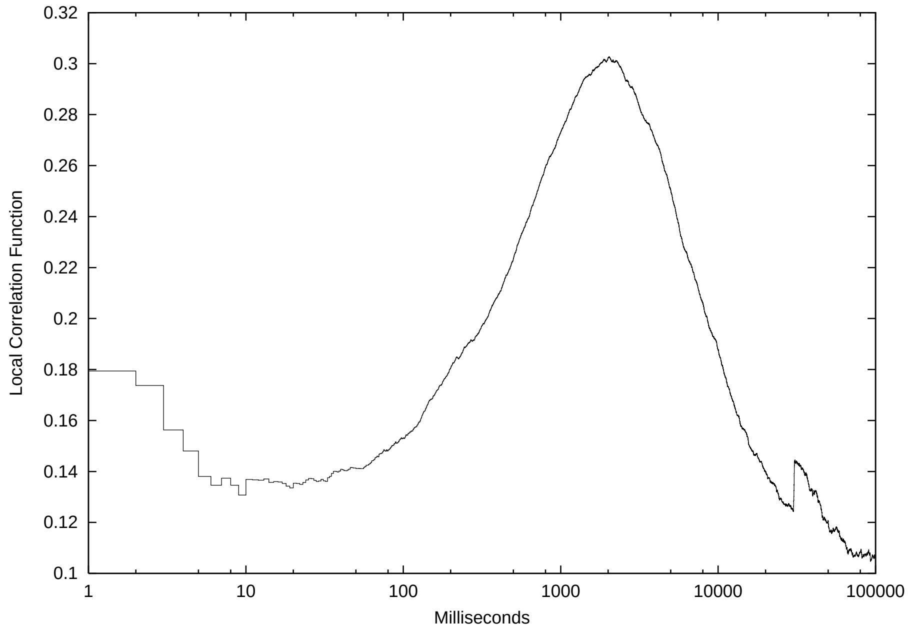




- Signifikanter Anstieg von Mittelwert und Varianz, dennoch nur geringfügig veränderte Verteilungsfunktion nach logarithmischer Transformation.
  $\Rightarrow$ **Verteilungstyp kann beibehalten werden, Anpassung der Parameter.**
- Grund für die Veränderung: keep-alive und grössere Datenmengen

Verbindungs-Zwischenankunftszeiten





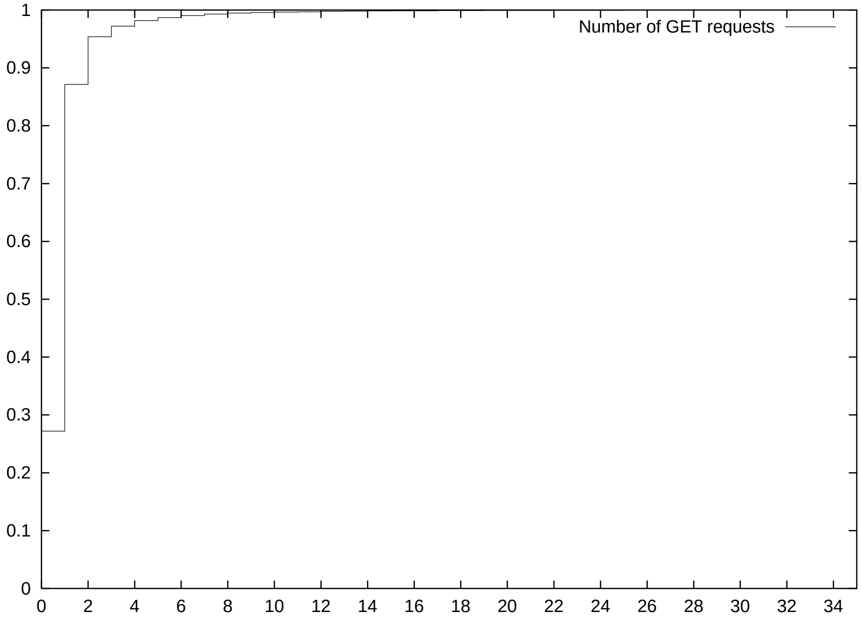
Verbindungs-Zwischenankunftszeiten(2)

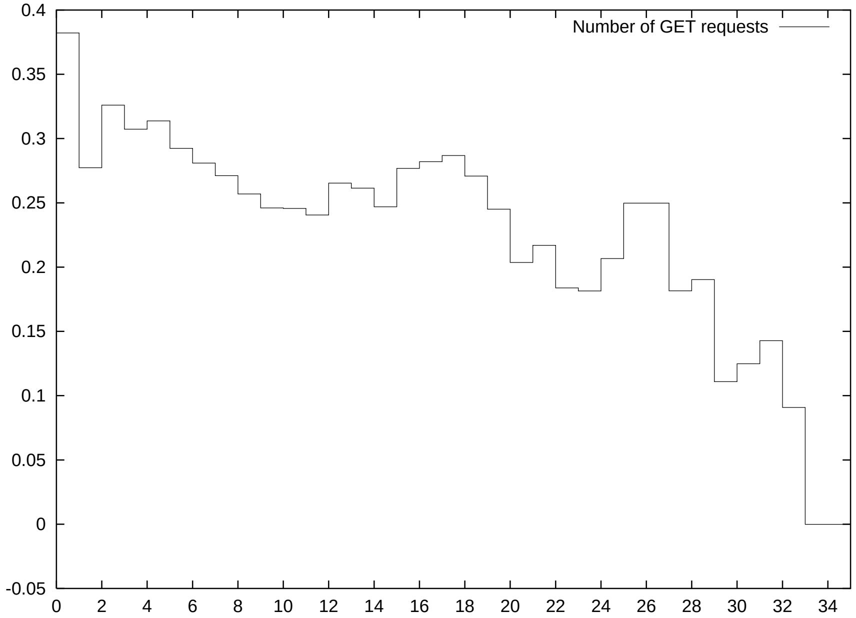
Klassifizierung von Zwischenankunftszeiten:

- Aufbau von parallelen Verbindungen beim Web-Request
- Lesezeiten, anschließender Wechsel zu einem anderen Server ($\Rightarrow$ Lokalitätsmodell)
- Lesezeiten, anschließender Aufbau von Verbindungen zum gleichen Server, da die vorhandenen Verbindungen zwischenzeitlich abgebaut wurden.
- Inaktive Zeiten des Benutzers

Gemeinsame Modellierung, da sich Zwischenankunftszeiten und Lesezeiten überlappen können bzw. parallel stattfinden.

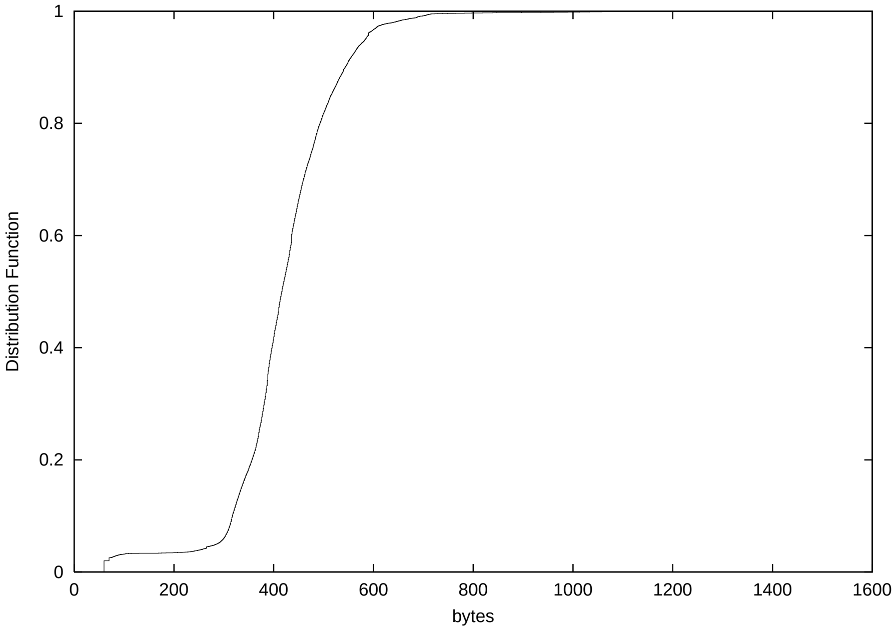
Anzahl der in einer Verbindung übertragenen Objekte

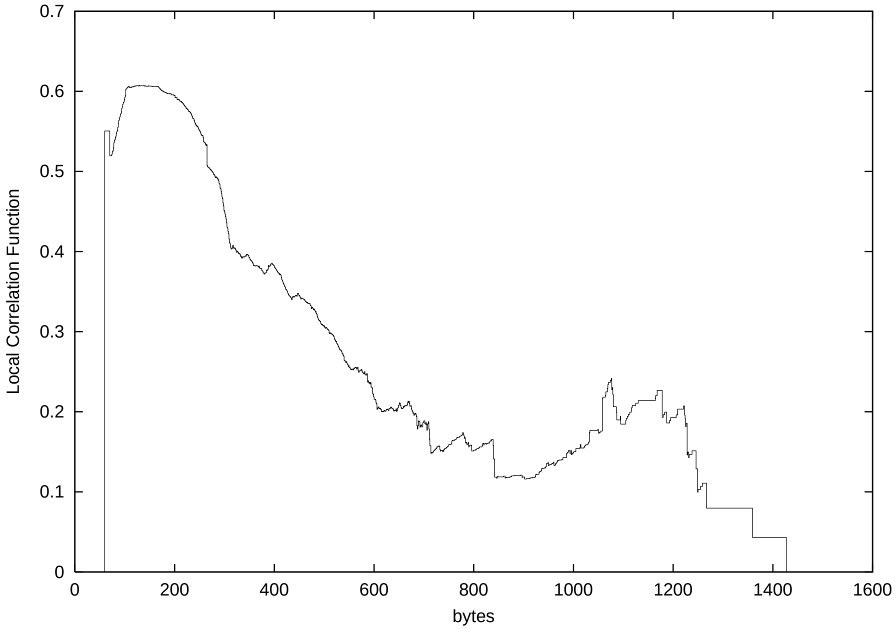




- Benutzung des keep-alive/persistent-connection Mechanismus deutlich erkennbar.
- starke Korrelation

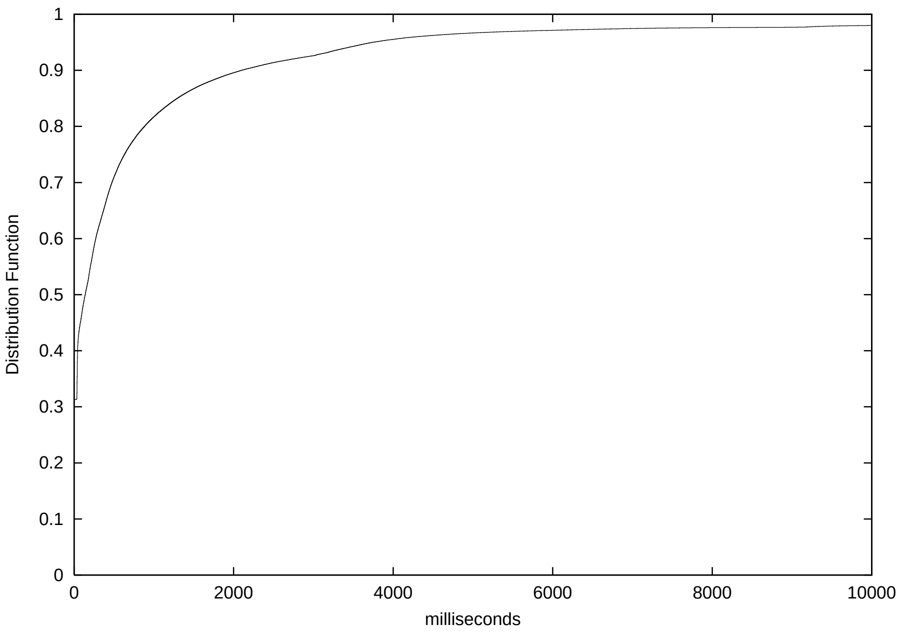
Paketgrößen der GET-Requests

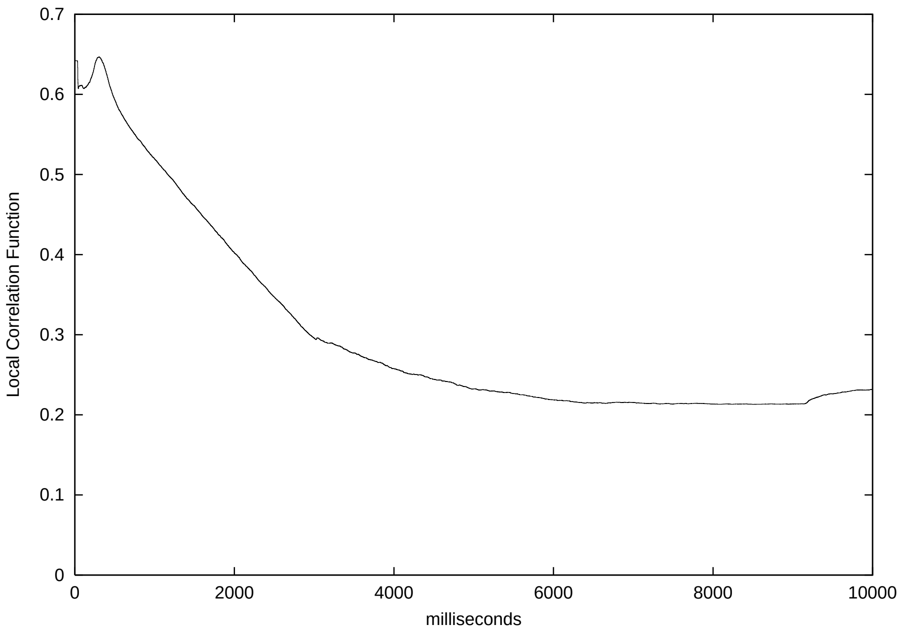




- Nur Requestmethode *GET* wird erfaßt.
- Einfluß komplexerer Serverstrukturen wird in großer Varianz erkennbar.

Verzögerungsintervall der GET-Requests





- Deutlicher Einfluß der Verzögerung sichtbar
  $\Rightarrow$ Modellierung dieses Effekts ist notwendig
- Mittelwert ca. 423 Sekunden, Varianz $2.322 * 10^{13}$

4. Modellierung

- **Unterteilung in Benutzermodell und HTTP-Modell auf der Client-Seite.**

- **Genaue Modellierung der Zeitpunkte von Verbindungsauf- und -abbau.** Bisherige Ansätze nahmen vereinfachend an, daß über eine TCP-Verbindung nur ein Objekt übertragen wurde.

- Die Möglichkeit der zeitlichen Überlappung einzelner Web Requests wurde integriert. $\Rightarrow$ **Parallele Browser-Instanzen und mehrere Benutzer unter einer IP-Adresse können simuliert werden.**

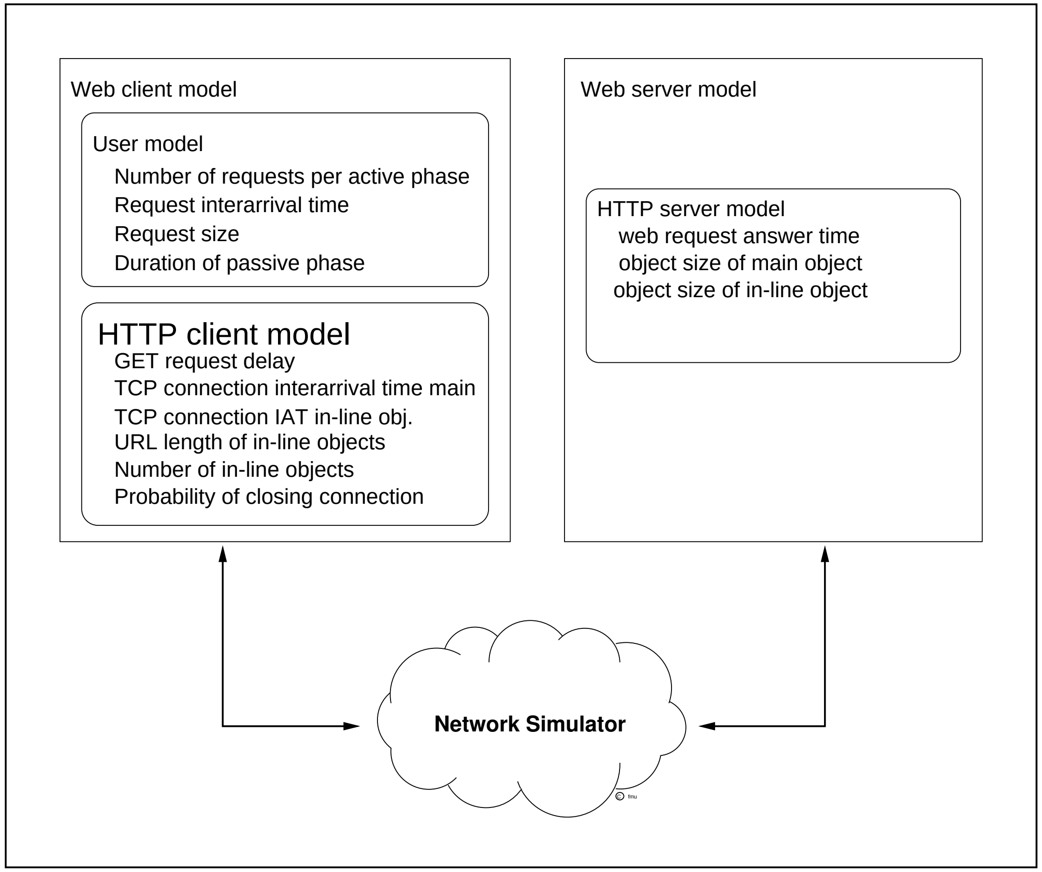
Web Request

Ein *Web Request* besteht aus dem Bezug von Informationen, die aus einer oder mehreren Internetseite(n) inklusive der eingebetteten Objekte bestehen. Per Definition wird ein Web Request durch Benutzeraktionen ausgelöst und das erste übertragene Objekt ist ein HTML-Dokument [2].

Kriterien zur Identifikation von Web Requests:

- Werden gleichzeitig mehrere WWW-Seiten angefordert, so bildet jede Anforderung einen eigenen Web Request. Ausnahme: die Seitenanforderungen werden durch eine einzige Aktion eines Benutzers ausgelöst.
- Automatisch weitergeleitete Seiten (z.B. *moved permanently*) bilden keinen neuen Web Request.
- Wird ein einzelnes Objekt, etwa ein Pdf-Dokument, eine Video-Datei etc. angefordert, wird diese Anforderung wie die eines eingebetteten Objekts innerhalb des vorhergehenden Web Requests behandelt.

Modellarchitektur



Features:

- Netzseitig können sowohl einzelne Web Server als auch Gateways simuliert werden (Unterschied: Verteilung der Antwortzeiten im Server-Modell).
- Die Steuerung eines vorhandenen TCP-Protokolls wird nachgebildet.
- Unterteilung der Simulationszeit in aktive und passive Phasen.

Benutzer-Modell

Im Benutzer-Modell sind diejenigen Parameter zusammengefaßt, die eindeutig mit Aktionen des Benutzers im direkten Zusammenhang stehen.

Das Benutzer-Modell wird durch folgende Parameter bestimmt:

- Anzahl der Web Requests in einer aktiven Phase
- Größe der Pakete, in denen die Anforderung des einleitenden HTML-Dokuments übertragen wird (entspricht der URL-Länge).
- Zwischenankunftszeiten der Web Requests
- Dauer der inaktiven Phasen

HTTP-Client-Modell

Im HTTP-Client-Modell wird der Web-Browser nachgebildet. Anforderungen des Benutzer-Modells werden entgegengenommen und bearbeitet.

Das Client-Modell benötigt folgende Parameter:

- Verzögerung zwischen dem Aufbau einer Verbindung und dem Versenden des ersten Web-Requests
- Zwischenankunftszeiten der TCP-Verbindungen
- Anzahl der eingebetteten Objekte eines Web-Requests
- URL-Längen der eingebetteten Objekte
- Wahrscheinlichkeit, daß eine Verbindung nach dem Transfer des gerade übertragenen Objekts geschlossen wird.

HTTP-Server-Modell

Nachbildung des Verhaltens von Web-Servern und möglicherweise eines Teils der Übertragungsstrecke, falls diese innerhalb des Lastmodells abstrahiert wird.

Parameter:

- Größen von Hauptobjekten und eingebetteten Objekten
- Antwortzeiten auf URL-Anfragen

Zusammenfassung

- Durch Messung und Auswertung aktuellen Internetverkehrs wurde gezeigt, daß bisher häufig verwendete Modelle wichtige Verkehrscharakteristika nicht abbilden.
- Das Verhalten moderner Web Browser im Zusammenspiel mit zur Zeit verwendeter Serversoftware wurde genau untersucht, um daraus Vorschläge für ein generisches, von spezifischer Software unabhängiges Modell abzuleiten.

Zukünftige Aufgaben:

- Auswahl geeigneter (korrelierter) Generatoren als Basis des Modells
- Vollständige Bestimmung aller Modellparameter
  $\Rightarrow$ weitere Messungen sind erforderlich (inkl. Korrelation höherer Ordnung)
- Implementierung des Modells
- Vergleich des generierten Verkehrs mit Messungen realen Verkehrs

Literatur

[1] M. Williamson, C. Arlitt: A Synthetic Workload Model for Internet Mosaic Traffic, Technical Report, Department of Computer Science, University of Saskatchewan, 1995.

[2] H. Choi, J. Limb: A Behavioral Model of Web Traffic. Technical Report, Georgia Institute of Technology, 1999.

[3] F. Schreiber, C. Görg: Stochastic Simulation: a simplified LRE-Algorithm for Discrete Random Sequences. AEÜ, Vol. 50, pp. 233-239, 1996.

[4] T. Irnich: Measuring and Modelling WWW Traffic Workload Characteristics in Access Networks. Student Project Thesis, Comnets, RWTH Aachen, July 2000.
Erhältlich zum download: *http://www.comnets.rwth-aachen.de/~tim*