Konzeption und Entwicklung eines echtzeitfähigen Lastgenerators für Multimedia-Verkehr in IP-basierten Rechnernetzen

Vortragender: Andrey Kolesnikov

1. Motivation für Lastgeneratoren (LG)
2. Eine formale Methode zur Lastspezifikation
3. Ausgangssituation (bestehende Architektur und Werkzeuge)
4. Entwurf für einen echtzeitfähigen Lastgenerator
5. Aspekte der Realisierung des LG
6. Experimente zur Ermittlung der Leistungsfähigkeit und der Präzision des LG
7. Ausblick

1. Motivation für Lastgeneratoren

- Wachsende Komplexität moderner Kommunikations- und Informationssysteme
- Steigende Bedeutung von Leistungsanalysen und Verhaltensprognosen *unter verschiedenen Lastsituationen / Belastungsniveaus* für diese Systeme
- Möglichkeiten zur Herstellung verschiedener Lastszenarien:

  a) Einsatz von ***realen Lasten*** (Aufträge von realen Anwendungen)

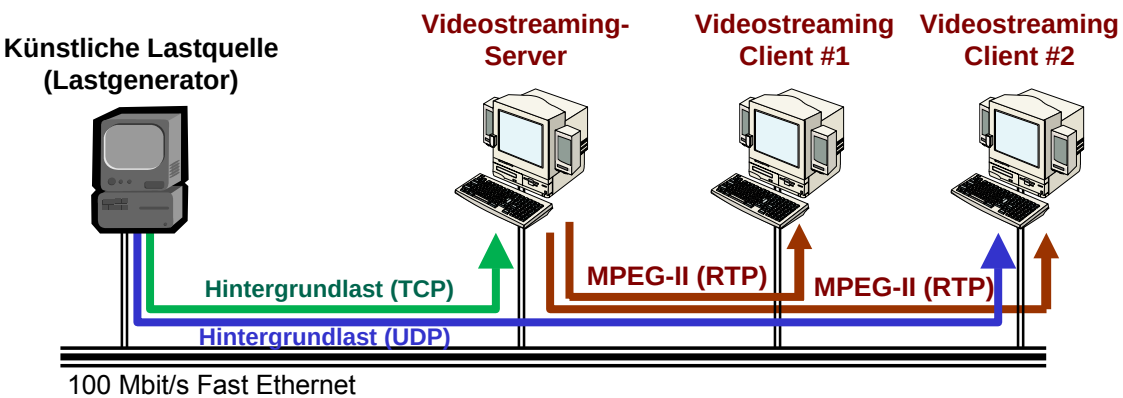
  b) Einsatz von ***synthetischen (künstlichen) Lasten***

  - basierend auf einem *Lastmodell (modellbasierte Lastquellen)*
  - *Lastmodell* abgeleitet aus Messungen an einem realen System oder einer realen Anwendung

  + Herstellung von speziellen Lastprofilen (*Reproduzierbarkeit*)
  + vielfältige *Parametrisierungsmöglichkeiten* (*Flexibilität*)
  + *Skalierbarkeit* (z.B. in der Anzahl der Benutzer, Ströme)
  + Geringerer *Aufwand* für die Installation der Testumgebung, Experimentvorbereitung, etc.

Rolle der Lastmodellierung, -spezifikation und -generierung:

- Geignete ***Methoden zur Lastmodellierung und Lastspezifikation*** sind von hoher Bedeutung für die Informatik / Telematik
- Wachsender Bedarf nach universellen und flexiblen ***Werkzeugen zur Lastgenerierung*** an verschiedenen (Dienst-)Schnittstellen in Informations- und Kommunikationssystemen
- *Dr. Ratul Mahajan (Microsoft Research)*: “The area of generating synthetic traffic for experiments and testing is an important one ...” *SIGCOMM’06 (Pisa, Sept. 2006)*



Ein mögliches Einsatzszenario: Untersuchung der Videostreaming-Qualität bei verschiedenen Hintergrundlasten

1. Weshalb Lastgenerierung in Echtzeit?

Heutige Dienstschnittstellen (z.B. *TCP*) häufig ***mit temporären Blockierungssituationen*** (für Benutzer) verbunden

- Berücksichtigung der Systemabhängigkeit bei Lastgenerierung unverzichtbar

Ausgangssituaton:

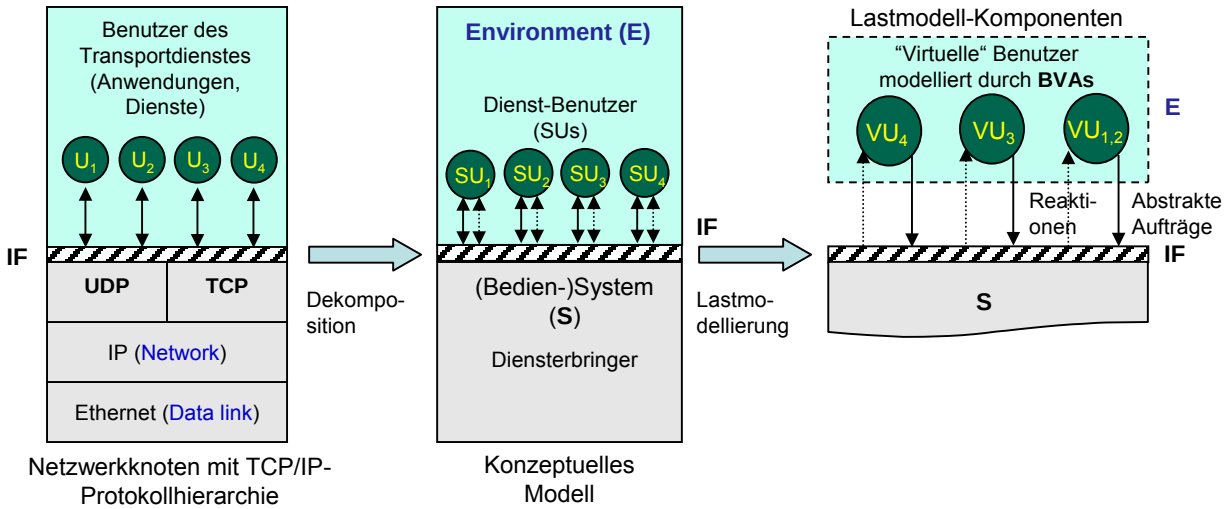
Existierende Lastgeneratoren (z.B. Harpoon [SoB04], BRUTE [BGP05]) unabhängig vom Systemzustand

Ziel der vorliegenden Diplomarbeit:

Entwurf und Realisierung eines Lastgenerators für systemzustandsabhängige Auftragsgenerierung

- Anspruchsvolle Echtzeitanforderungen!

2. Eine formale Methode zur Lastspezifikation ([Wo99, Co06])



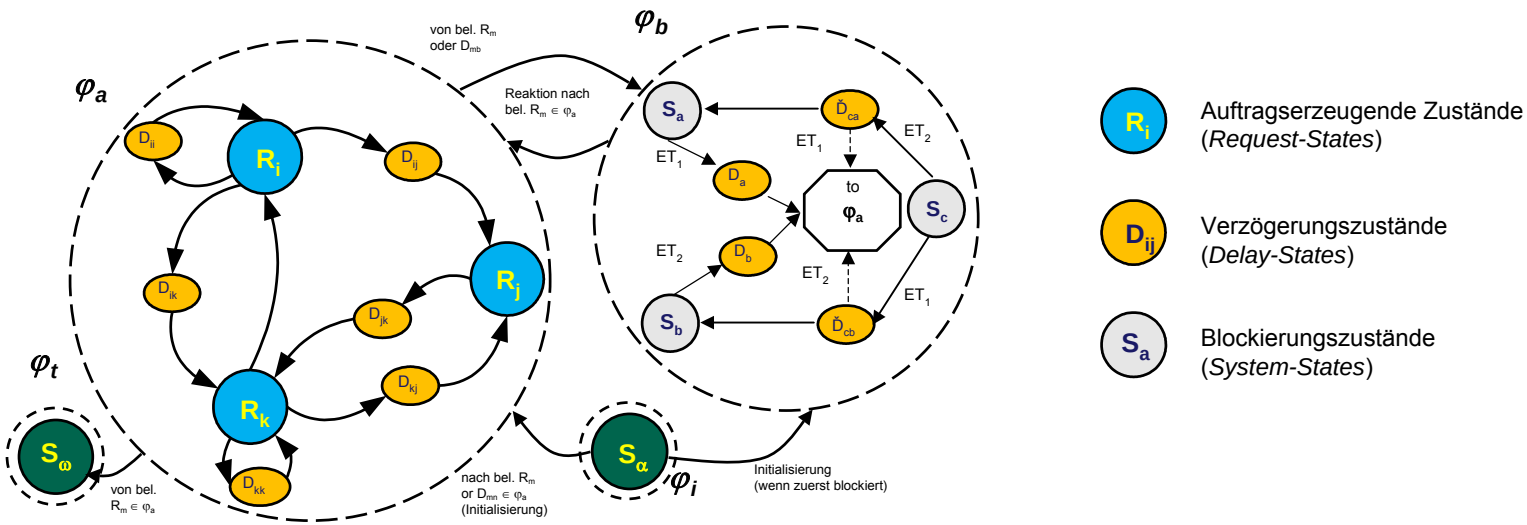
Last (*offered workload*) L = L(*E*, *S*, *IF*, *T*):

Die gesamte Sequenz der Aufträge, die von Seiten der Umgebung *E* an das Bediensystem *S* an einer wohldefinierten Schnittstelle *IF* in einem Zeitintervall *T* übergeben werden:
L = (t_i, A_i), $0 \le t_i \le T$, i = 1,2,…,n ($t_i \in$ R: Übergabezeitpunkt des Auftrags A_i; $t_i \le t_j$ für i<j)

Schritte der formalen Lastspezifikationstechnik:

(1) Festlegung einer (Ziel-)*Schnittstelle IF* für Lastmodellierung (z.B. *HTTP*, *TCP*, *IP*, …)
(2) Spezifikation der möglichen Aufträge durch *abstrakte Auftragstypen* und ihre *Attribute*
(3) Spezifikation der möglichen *Sequenzen* abstrakter Aufträge
(mittels *Benutzerverhaltensautomaten*, *BVA*)
(4) Spezifikation der *Werte der Auftragsattribute* und der *Übergabezeitpunkte* der Aufträge.

2. Eine formale Methode zur Lastspezifikation ([Wo99, Co06])



Ein ***Benutzerverhaltensautomat*** (***BVA***) (engl. *user behaviour automaton*, *UBA*), U = $\{\varphi_i, \varphi_a, \varphi_b, \varphi_t, T_\varphi\}$ ist ein erweiterter endlicher Automat bestehend aus der Menge der *Makrozustände* $\varphi = \{\varphi_i, \varphi_a, \varphi_b, \varphi_t\}$ und der Menge T_φ der Transitionen (Übergänge) zwischen diesen Makrozuständen.

3. Ausgangssituation: existierende Werkzeuge

Werkzeuge zur Lastspezifikation (abstrakte Aufträge):

- Werkzeug **LoadSpec** zur Spezifikation und Generierung abstrakter Aufträge an *IF*
- Einsatz in der Lehre zur Veranschaulichung der Lastbeschreibungstechnik mit BVAs (für Informatik-Studierende im Hauptstudium), z.B. PCM-codierte Audio- und MPEG-codierte Videoströme ([FHW04])
- Werkzeug **LoadTrafo** zur Modellierung der lasttransformierenden Wirkung des Bediensystems (Veränderung der Auftragslängen oder des Ankunftsprozesses)

Werkzeuge zur Lastgenerierung (reale Aufträge):

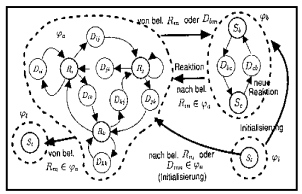
- Werkzeug **UniLoG** ([CK05]) zur Spezifikation und Generierung von abstrakten und realen Aufträgen (mithilfe von UDP- und TCP-Adapter)
- Graphische Spezifikation des Benutzermodells (BVA) und dessen Parametrisierung (PBVA)
- Generierung von abstrakten Aufträgen (GAR)
- Generierung von realen Aufträgen an IF durch einen schnittstellenspezifischen Adapter (ADAPT)

3. Ausgangssituation: bestehende Architektur UniLoG

Experimentator

GUI

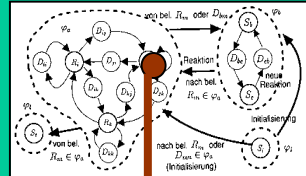
process



PBVA Konstruktions-komponente

process

Generator (GAR)



BVA Modellbank

Parameter Bank (PB)

abstrakte Aufträge

process

Adapter (ADAPT)

```
while (pop(OQ))  {
  Do {
    Now = GetTimeOfDay();
    } while (Now < t1);
  sendto(134.100.15.171,
         1024);     }
```

reale Aufträge

Netzschnittstelle z.B. UDP oder TCP

BVA:	Benutzerverhaltensautomat
PBVA:	parametrisierter BVA
GAR:	Generator for Sequences of Abstract Requests

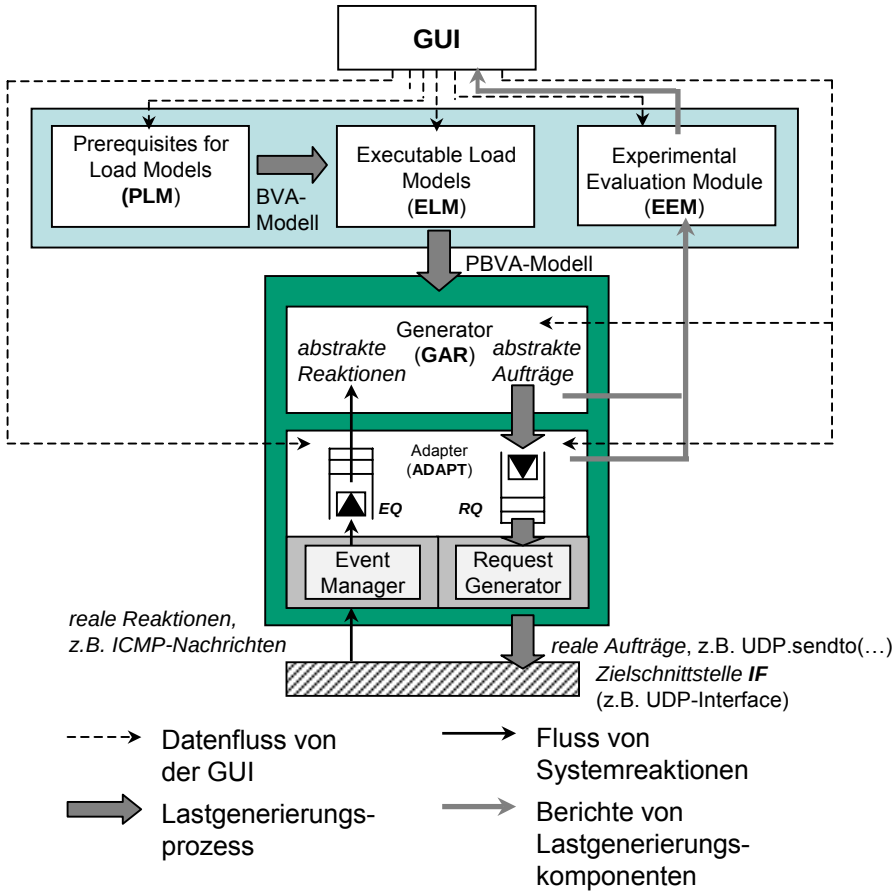
Besondere Herausforderung:

Unterstützung der Systemabhängigkeit im Lastgenerierungsprozess:

- Realisierung der netzbedingten Wartezustände des Benutzers
- Rückfluss von Nachrichten vom Adapter zum Generator erforderlich
- Zeitlich getrennte Generierung von abstrakten und realen Aufträgen nicht mehr möglich, im neuen Entwurf nebenläufig

Wünschenswerte Erweiterung:

Überlagerung von Auftragssequenzen mehrerer modellierter Benutzer (*BVA*) zur Erzeugung spezieller "Traffic-Mixe"



Beidseitige "Produzenten-Konsumenten"-Beziehung zwischen dem Generator und dem Adapter

Randbedingungen für den Entwurf der Teilkomponenten:

(1) Benutzer können in den netzbedingten Wartezuständen des BVA blockiert sein

(2) die Warteschlangen *RQ* und *EQ* haben beschränkte Kapazität

(3) alle Aufträge sind zeitkritisch (die spezifizierten Übergabezeitpunkte sind einzuhalten).

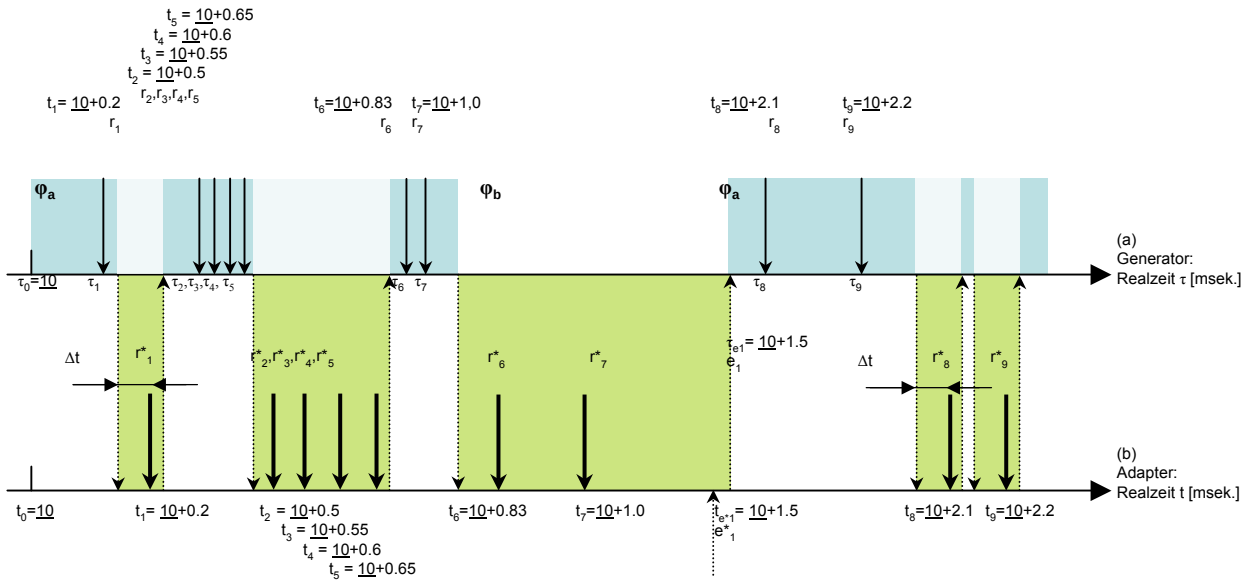
Zu berücksichtigende Faktoren:

- *Verarbeitungszeiten in den Komponenten*: alle an der Lastgenerierung beteiligten Komponenten und Prozesse benötigen zur Bearbeitung des nächsten generierten Auftrages eine bestimmte Verarbeitungszeit

- *betriebssystembedingte* Faktoren (Einfluss anderer Prozesse im Multitasking-Betrieb)

- *modellbedingte Faktoren*, z.B. stoßweises Auftragsaufkommen ("Bursts")

4. Sichere Kontrollübergabe zwischen dem Generator und dem Adapter



Ein Auftrag A wird als *dringend (urgent)* zum Zeitpunkt t_{NOW} bezeichnet, wenn bis zu seinem physikalischen Übergabezeitpunkt t_{NEXT} weniger als eine festgelegte Zeit Δt übrig bleibt:

$$(t_{NEXT} - t_{NOW}) < \Delta t$$

Möglichkeiten für die Modellierung von Systemreaktionen:

- **Lokales reaktives Modell:**

  Generierung der abstrakten Reaktionsnachrichten durch den Adapter gemäß den Spezifikationen in den *S-Zuständen* des BVA; Parametrisierung mittels Systemtraces.

- **Verteiltes reaktives Modell:**

  Generierung von abstrakten Reaktionsnachrichten gemäß einem analytischen oder simulativen Modell des Netzverhaltens, ausgeführt in einem Netzemulator (z.B. *NetEmu*, [Sch06]).

- **Hybride Methode:** Verarbeitung von realen Reaktionen an der Schnittstelle *IF*. Die Verfügbarkeit der entsprechenden Dienstprimitiven bzw. Funktionscodes zum Abruf von Statusinformationen wird vorausgesetzt (z.B. ICMP-Nachrichten "Destination Unreachable", "Time Exceeded", "Source Quench" etc.).

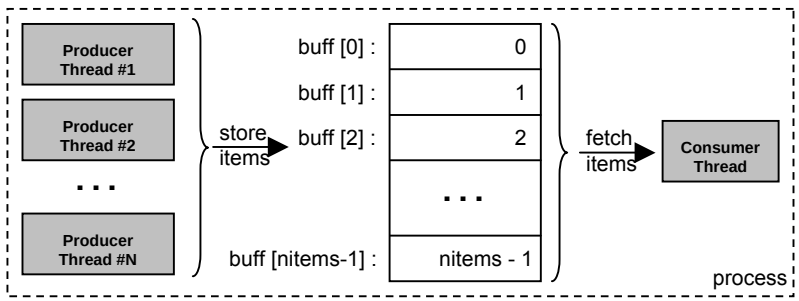
Besondere Herausforderung:

die exakte Einhaltung der Auslieferungszeitpunkte bei der Auftragsgenerierung

⇒ Auswahl geeigneter Synchronisations- und Uhrprimitiven ([MSDN])

Implementierung unter Windows mit Threads:

Producer-Consumer Problem ([Stevens 99])



- **N producer threads:** N lastgenerierende Benutzer (GAR_1,…,GAR_N)
- **1 consumer thread:** der *UDP*- oder der *TCP-Adapter*
- IPC zwischen GAR_i und dem *UDP- bzw. TCP-Adapter* über den "bounded buffer" (begrenzte Pufferkapazität) mithilfe von Semaphoren

6. Stand der Implementierung

Erste Pilotimplementierung LG:

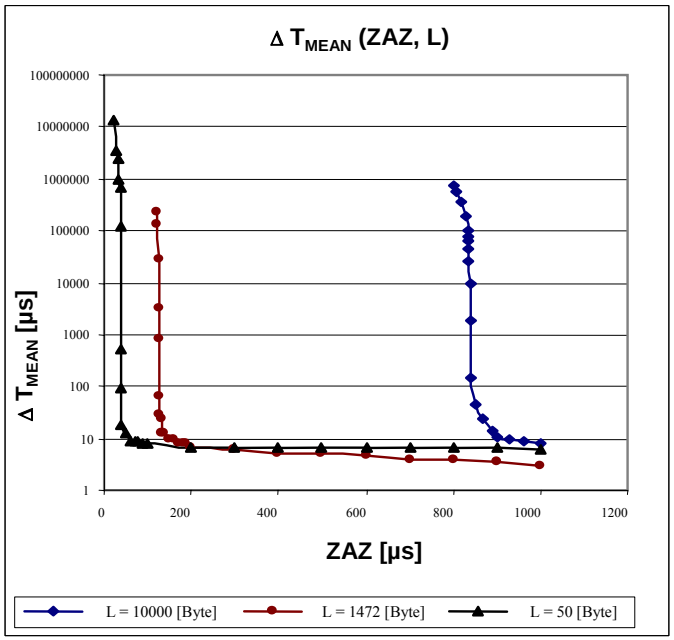
- alle GAR_i führen denselben PBVA aus
- TCP-Adapter und UDP-Adapter als Consumer-Threads
- TCP- und UDP-Lastsenken

Einstellbare Modellparameter:

- Lastgenerierungsdauer
- ZAZ der Aufträge: konstant bzw. exponentialverteilt
- Initialisierungszeit des Generators (bis alle Threads in ihrem I-Zustand)
- Auftragslänge, konstant (10000, 1472, 50 [Byte])
- Adressen (in der Transportschicht) des Senders und des Empfängers
- zu verwendender Adapter (*TCP* oder *UDP*)

Experimente zur Analyse der Leistungsfähigkeit und der Präzision des LG:
Wie nah an dem rechnerischen Übergabezeitlimit können die ZAZ der UDP- bzw. der TCP-Aufträge spezifiziert werden?
(3x Intel® Pentium® 4 CPU 2,40 GHz, 1,00 GB RAM, Intel PRO/100 NIC, Microsoft Windows XP Professional, Version 2002, Service Pack 2)

6. UDP-Adapter, CBR-Verkehr



$$\Delta T(r_i) = T_{IST}(r^*_i) - T_{SOLL}(r_i),\ \forall i = 1,2, ..., N$$

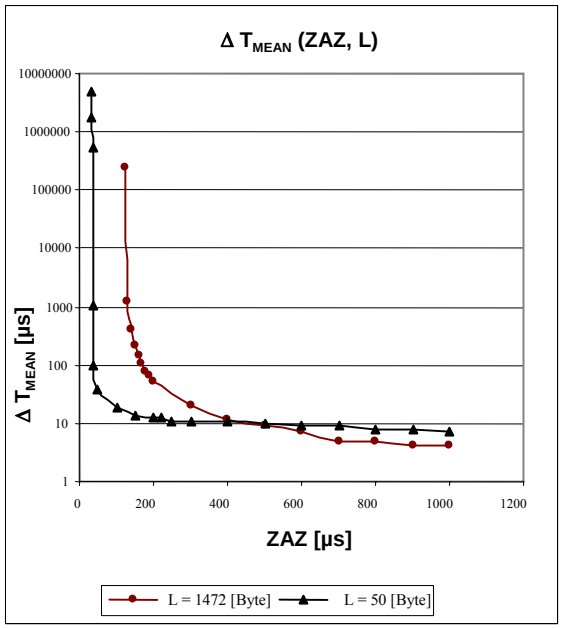
$T_{SOLL}(r_i)$: spezifizierter ÜZP des abstrakten Auftrags r_i (im Generator *GAR*)

$T_{IST}(r^*_i)$: faktischer ÜZP des entsprechenden UDP-Auftrags r^*_i im Adapter

$$\Delta T_{MEAN} = \overline{\Delta T} = \sum_{i=1}^{N} \frac{\Delta T(r_i)}{N}$$

	Experimentserie 1	Experimentserie 2	Experimentserie 3
Lastgenerierungsdauer [sec]	30	30	30
Auftragslänge [Byte]	10000	1472	50
Übergabezeitlimit (Fast Ethernet) [µs]	823.69	121.12	7.36
(konstante) ZAZ [µs]	2000 ÷ 835	500 ÷ 123	400 ÷ 36
98% der Werte $\Delta T < 10$[µs]	bei ZAZ > 1500[µs]	bei ZAZ > 140[µs]	bei ZAZ > 41[µs]
95% der Werte $\Delta T < 10$[µs]	bei ZAZ > 875[µs]	bei ZAZ > 126[µs]	bei ZAZ > 38[µs]

6. UDP-Adapter, VBR-Verkehr



ZAZ: *negativ-exponentiell* (λ), $E[ZAZ] = 1/\lambda$

$\Delta T(r_i) = T_{IST}(r^*_i) - T_{SOLL}(r_i), \forall i = 1,2, ..., N$

$T_{SOLL}(r_i)$: spezifizierter ÜZP des abstrakten Auftrags r_i (im Generator *GAR*)

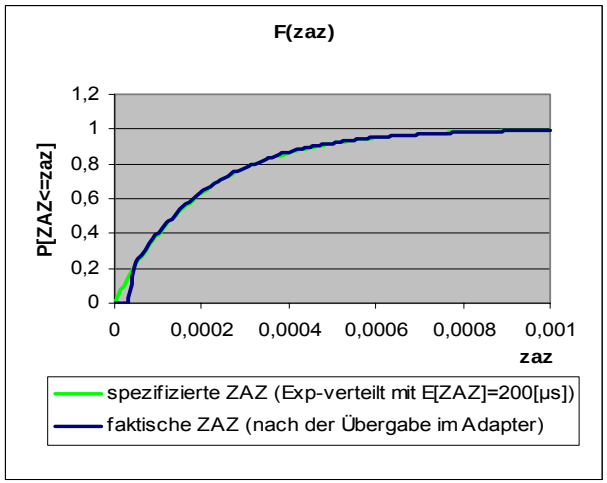
$T_{IST}(r^*_i)$: faktischer ÜZP des entsprechenden UDP-Auftrags r^*_i im Adapter

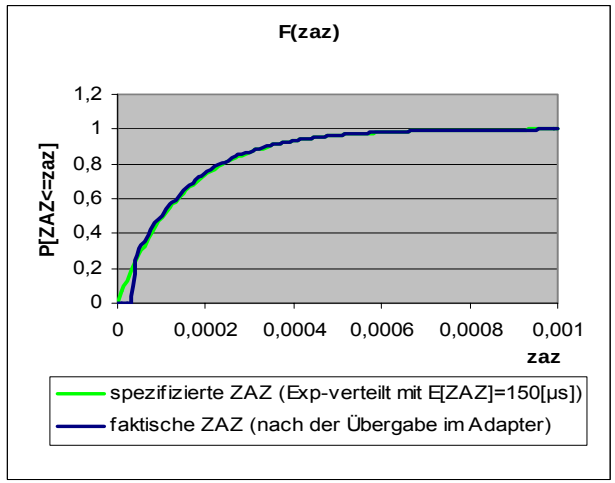
$$\Delta T_{MEAN} = \overline{\Delta T} = \sum_{i=1}^{N} \frac{\Delta T(r_i)}{N}$$

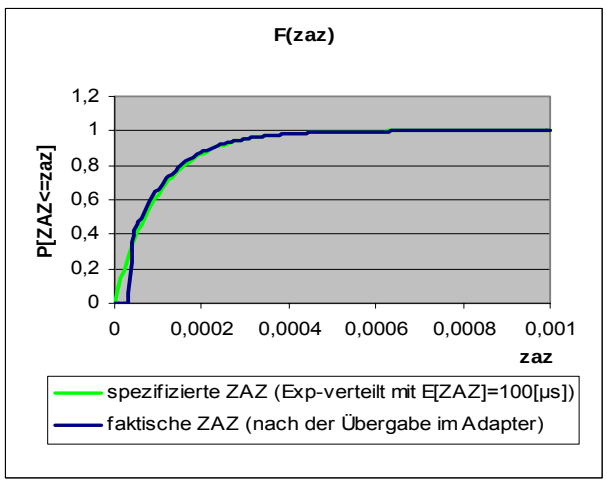
	Experimentserie 4	Experimentserie 5
Lastgenerierungsdauer [sec]	30	30
Auftragslänge [Byte]	1472	50
Übergabezeitlimit (Fast Ethernet) [µs]	121.12	7.36
E[ZAZ] [µs]	500 ÷ 123	400 ÷ 36
(98% der Werte ΔT) < (10% von E[ZAZ])	bei ZAZ > 600[µs]	bei ZAZ > 280[µs]
(95% der Werte ΔT) < (10% von E[ZAZ])	bei ZAZ > 400[µs]	bei ZAZ > 240[µs]

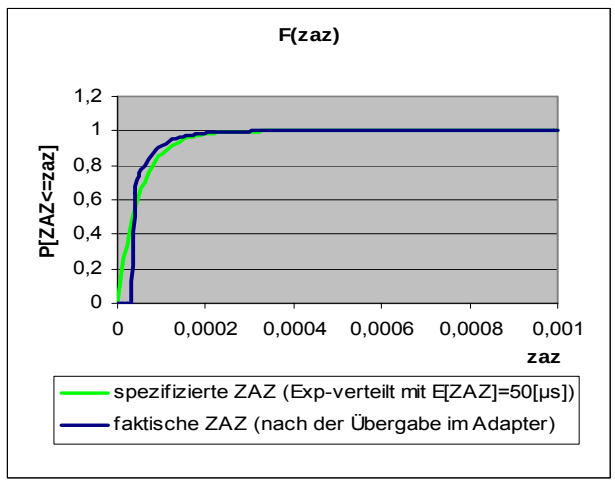
6. UDP-Adapter, VBR-Verkehr

spezifizierte Verteilung der ZAZ: negativ-exponentiell mit E[ZAZ]=1/λ
tatsächlich erzielte (Häufigkeits-)Verteilung der ZAZ im Adapter

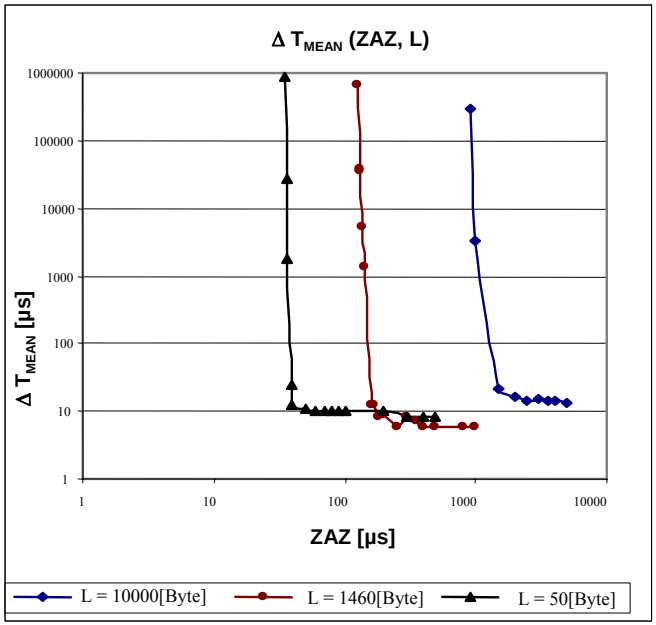








6. TCP-Adapter, CBR-Verkehr



$\Delta T(r_i) = T_{IST}(r^*_i) - T_{SOLL}(r_i), \forall i = 1,2, ..., N$

$T_{SOLL}(r_i)$: spezifizierter ÜZP des abstrakten Auftrags r_i (im Generator *GAR*)

$T_{IST}(r^*_i)$: faktischer ÜZP des entsprechenden TCP-Auftrags r^*_i im Adapter

$$\Delta T_{MEAN} = \overline{\Delta T} = \sum_{i=1}^{N} \frac{\Delta T(r_i)}{N}$$

	Experimentserie 1	Experimentserie 2	Experimentserie 3
Lastgenerierungsdauer [sec]	30	30	30
Auftragslänge [Byte]	10000	1460	50
Übergabezeitlimit (Fast Ethernet) [µs]	823.69	122.08	7.36
(konstante) ZAZ [µs]	10000 ÷ 900	1000 ÷ 125	400 ÷ 40
95% der Werte $\Delta T < 10$[µs]	bei ZAZ > 10[ms]	bei ZAZ > 250[µs]	bei ZAZ >190[µs]
95% der Werte $\Delta T <$ (10% von E[ZAZ])	bei ZAZ > 3.5[ms]	bei ZAZ > 190[µs]	bei ZAZ >100[µs]

7. Ausblick

Aktuelle Forschungsarbeiten im UniLoG-Umfeld:

- Untersuchungen der Qualität von Video-Streaming (H.264/AVC) in WLANs (*UniLoG* zur Modellierung verschiedener Hintergrundlasten)
- Realisierung eines IP-Adapters *UniLoG.IPv4* zur Lastgenerierung an IP-Schnittstellen; bereits existent: UDP-, TCP-, HTTP-Adapter
- Erweiterung der BVA-Spezifikation zur Unterstützung von reaktiven Lastmodellen (z.B. durch Einführung von *bedingten Zustandsübergängen*)

Weitere geplante Forschungsaktivitäten:

- Lastmessungen, -charakterisierung und -modellierung für den HTTP-Verkehr / Web 2.0-Anwendungen (z.B. im Master-Projekt), Einsatz in *UniLoG.HTTP*
- Einbindung von *UniLoG* in eine Umgebung zur verteilten Lastgenerierung
- Integration von *UniLoG* in einen Netzsimulator (z.B. *ns-2*) bzw. Kopplung mit einem Netzemulator (z.B. *NetEmu*)
- Realisierung und Einbindung von Bausteinen zur Lasttransformation (z.B. Token-Bucket, "Sliding Window"-Mechanismus von TCP, etc.).

Vielen Dank für Ihre Aufmerksamkeit !

Und ein großes Dankeschön für die Einladung zum Vortrag auf der "Echtzeit 2008" !