Vermessung von Netzwerkcharakteristiken über Mobilfunkverbindungen

Bachelorarbeit

von

Niklas Först

aus

Leverkusen

vorgelegt am

Lehrstuhl für Rechnernetze und Kommunikationssysteme
Prof. Dr. Martin Mauve
Heinrich-Heine-Universität Düsseldorf

April 2014

Betreuer:
Norbert Goebel, M. Sc.

Zusammenfassung

Das Ziel dieser Bachelorarbeit, war die Implementierung eines Ansatzes zur Latenz, Daten- und Dropratenmessung. Um self-induced-congestion, welche in der Messsoftware einer vorherigen Arbeit auftrat, vermeiden zu können, wurde für die Messung ein ”Ping-Pong”-Prinzip genutzt. Hierbei wird abwechselnd jeweils nur in eine der Kommunikationsrichtung gemessen.

Zur Evaulierung der Messsoftware wurden Messungen unter Laborbedingungen durchgeführt. Es wurden Daten- und Droprate der Verbindung angepasst, um die Funktionsfähigkeit der Messsoftware zu überprüfen. Weiterhin wurden Timer für Paketverluste implementiert und durch weitere Messungen untersucht, inwiefern deren Parameter sinnvoll zu wählen sind.

Danksagung

Ich möchte allen Menschen danken, die mir bei dieser Bachelorarbeit beigestanden haben. Dazu gehört meine Familie, die mich die ganze Zeit über, im Alltag und generell Immer unterstützt hat. Weiterhin möchte ich den Leuten, welche die Zeit auf sich genommen haben, die Arbeit noch einmal zu lesen und zu korrigieren, danken. Ein großer Dank geht auch an Norbert Goebel, der mir als Betreuer mit Rat und neuen Ideen immer wieder zur Seite gestanden hat.

Inhaltsverzeichnis

Abbildungsverzeichnis **ix**

Tabellenverzeichnis **xi**

Listings **xiii**

1 Einleitung **1**

1.1 Motivation 1

1.2 Überblick 2

2 Grundlagen **3**

2.1 Latenz-, Dropraten- und Datenratenberechnung 3

2.2 Zeitsynchronisation 4

3 Die Messsoftware **7**

3.1 Konzept zur Latenz- und Datenratenmessung 7

3.2 Aufbau eines Messpakets 8

3.3 Logging 9

3.4 Threads 10

3.5 Timer 11

4 Evaluation **15**

4.1 Messungen unter Laborbedingungen 15

4.1.1 Anpassung von Paketverlust 15

4.1.2 Überprüfung mit iperf-Messungen 16

4.2 NTP-Konfiguration 17

4.3 Probemessungen für Datenrate 18

4.3.1 Probemessungen mit angepasstem Paketverlust 24
4.4 Bestimmung der Timer-Variablen 26

5 Zusammenfassung und Ausblick **31**

Literaturverzeichnis **33**

Abbildungsverzeichnis

3.1 ”Ping-Pong”-Prinzip 8
3.2 Thread Zusammenhang 10
3.3 Paket-Timer-Prinzip 12
3.4 Packettrain-Timer-Prinzip 13

4.1 Datenraten mit fester Paketgröße 20
4.2 Fehlerbalken mit fester Paketgröße 20
4.3 Datenraten mit fester Paketanzahl 21
4.4 Fehlerbalken mit fester Paketanzahl 21
4.5 kumulative Verteilung für Mesungen mit fester Paketgröße 22
4.6 kumulative Verteilung für Mesungen mit fester Paketanzahl 23
4.7 Datenrate bei gleichbleibender Packettraingröße 23
4.8 Datenratenmessung über einen Zeitraum von 600s 24
4.9 Dropmessung bei 1000x1250 Bytes 25
4.10 Datenratenmessung bei 1000x1250 Bytes 25
4.11 Verfrühte Paket-Timeouts bei fester Messdauer 27
4.12 Empfangene Packettrains bei fester Messdauer 27
4.13 verfrühte Packettrain-Timeouts bei fester Messdauer 28

Tabellenverzeichnis

4.1 Variation der Messparameter 19

Listings

2.1 Beispiel einer ntp.conf-Datei 5

3.1 Log-Datei 9

4.1 Iperf Server 16
4.2 Iperf Client 17
4.3 NTP-Konfiguration Server 17
4.4 NTP-Konfiguration Client 18

Kapitel 1

Einleitung

1.1 Motivation

Sicherheit im Straßenverkehr und dessen Effizienz sind in unserer heutigen Zeit wichtige Themen für eine Vielzahl von Interessengruppen. So fließen viel Geld und Ressourcen der Fahrzeugindustrie in die Entwicklung von verschiedensten Systemen, die diese Sicherheit bieten und die Effizienz steigern können. Ein vielversprechender Zweig ist hierbei die Fahrzeug-zu-Fahrzeug-Kommunikation (C2C), bei der drahtlose Kommunikationstechnologie in Fahrzeuge eingebaut wird, um einen Austausch von verkehrsrelevanten Informationen zwischen zwei Fahrzeugen zu gestatten. Hierbei konkurriert als Übertragungstechnologie WLAN 802.11p mit klassischem infrastrukturbasiertem Mobilfunk. Um die Realisierbarkeit von Fahrzeug-zu-Fahrzeug-Anwendungen über Mobilfunk festzustellen, entwickelt der Lehrstuhl für Rechnernetze ein Verfahren zur Simulation von Mobilfunknetzwerken auf Basis von Messungen.

In einer vorangegangen Arbeit wurde ein Messframework für Mobilfunkcharakteristiken, wie Latenz, Drop- und Datenrate, für x86 Systeme in C/C++ entwickelt, welches zeitgleich in beide Kommunikationsrichtungen Messungen vornimmt. Diese Vorgehensweise führt allerdings zu Messproblemen durch self-induced-congestion, bei starker Abnahme der verfügbaren Datenrate.

Daher ist das Ziel dieser Bachelorarbeit die Implementierung und Untersuchung eines abgewandelten Messverfahrens, bei dem keine zeitgleiche Kommunikation in beiden Richtungen stattfindet, sondern abwechselnd jeweils nur in eine Kommunikationsrichtung gesendet wird. Dies ist ein Versuch die self-induced-congestion, wie sie im vorherigen Messframework auftrat, zu vermeiden. Hierbei soll die Messsoftware Latenz, Drop- und Datenrate bestimmen können. Anhand von abschließend durchgeführten Messungen soll untersucht werden, wie die nötigen Parameter der Software gewählt werden müssen, um gute Messergebnisse erzielen zu können. Hierzu zählen Paketgröße, Paketanzahl und Timer für Paketverlust.

1.2 Überblick

In Kapitel 2 werden die Grundlagen für die Messungen beschrieben. Hierzu zählen Berechnungen von Latenz, Drop- und Datenrate, sowie die Synchronisation von Systemzeiten von mehreren Rechner, welche für die Bestimmung der Latenz notwendig ist.

Kapitel 3 befasst sich mit den grundlegenden Konzepten der Messsoftware, sowie dem Aufbau der einzelnen Elemente der Messsoftware und der Timer.

In Kapitel 4 wird die Messsoftware evaluiert. Hierbei werden unter Laborbedingungen Daten- und Dropratenberechnung überprüft und die Timer betrachtet.

Kapitel 2

Grundlagen

2.1 Latenz-, Dropraten- und Datenratenberechnung

Die Aufgabe des Messprogramms ist die Berechnung von Latenz, Daten- und Droprate für eine Netzwerkverbindung, speziell auch im Mobilfunk. Dazu bedienen wir uns folgenden Konzepts zur Berechnung dieser Größen.

Es werden n Pakete mit bekannter Länge s_p in einem Packettrain back-to-back losgeschickt. Dabei wird jedes Paket innerhalb eines Packettrains mit einer Paketnummer gekennzeichnet. Außerdem bekommt jedes einen Sendezeitstempel t^x_{send}. Bei Empfang auf der Gegenseite wird wiederum jedes dieser Pakete mit einem Empfangszeitstempel t^x_{recv} versehen. Weiterhin wird festgehalten, welches das Erste und Letzte angekommene Paket ist und wie viele Pakete eines Packettrains generell angekommen sind. Somit haben wir alle Informationen zusammengetragen, um Latenz, Daten- und Droprate bestimmen zu können. Die Droprate kann mit Hilfe der Formel

$$droprate = 1 - \frac{n'}{n} \tag{2.1}$$

berechnet werden, wobei n' die Anzahl der tatsächlich erhaltenen Pakete darstellt. Weiterhin kann mit Hilfe von Sendezeitstempel t^x_{send} und Empfangszeitstempel t^x_{recv} die Latenz für ein Paket berechnet werden.

$$delay = t^x_{\text{recv}} - t^x_{\text{send}} \tag{2.2}$$

Die Datenrate A wird mit der Formel

$$A = \begin{cases} \frac{s_p * (last - first)}{t^{recv}_{last} - t^{recv}_{first}} & n' \geq 1 \\ \frac{s_p}{t^{recv}_{last} - t^{send}_{last}} & n' = 1 \\ NaN & \text{sonst} \end{cases} \tag{2.3}$$

berechnet, wobei t^{recv}_{first} den Empfangszeitstempel des ersten erhaltenen Pakets des Packettrains darstellt und t^{recv}_{last} den des zuletzt erhaltenen Pakets, bei t^{send}_{last} entsprechend. Weiterhin stellen $first$ und $last$ die Paketnummern des zuerst bzw. zuletzt erhaltenen Paketes dar.(vergleiche [GKMG14])

2.2 Zeitsynchronisation

Um die Sende- und Empfangszeitstempel für die Latenzberechnung sinnvoll nutzen zu können, müssen für die Kommunikationspartner des Messverfahrens die Systemzeiten synchron sein. In Computern erhöht ein eingebauter Taktgeber mit einer bekannten Frequenz den Wert eines Registers, um so die Zeit bestimmen zu können. Oft arbeiten diese Taktgeber jedoch nicht mit genau dieser angegebenen Frequenz oder werden durch Umwelteinflüsse, wie beispielsweise Temperaturschwankungen beeinträchtigt. Daher ist es notwendig die Uhren der einzelnen Systeme zu synchronisieren.

Für eine solche Synchronisation wurde für drahtgebundene Netzwerke das *Network Time Protocol (NTP)* entwickelt [Mil85, Mil88, Mil92, MMBK10]. Das Network Time Protocol Project stellt dazu eine Implementierung zur Verfügung, welche die Synchronisation zu Zeitservern im Internet ermöglicht. Weiterhin können auch in lokalen Netzwerken Zeitserver eingerichtet werden, mit denen die anderen Systeme synchron gehalten werden können. Um die Entfernung solcher Zeitserver von einer sehr genauen exter-

nen Zeitquelle, wie beispielsweise einem GPS-Empfänger, zu beschreiben, besitzt jeder Zeitserver ein sogenanntes Stratum-Level. So hat ein Server, der direkt an eine externe Zeitquelle angeschlossen ist ein Stratum von 1, ein Server der einen Stratum-1-Server als Referenz nutzt ein Stratum-Level von 2 und so fortlaufend. Die Konfiguration sowohl von NTP-Server als auch Client findet in der NTP-Konfigurations-Datei (*ntp.conf*) des jeweiligen Systems statt, was anhand von Listing 2.1 beispielhaft erklärt wird.

```
### Server :/etc/ntp.conf ######

#Abweichung
driftfile /var/lib/ntp/ntp.drift

# NTP-Server
server 0.pool.ntp.org burst prefer
server 1.pool.ntp.org iburst
server 127.127.1.0
fudge  127.127.1.0 stratum 10

#Zugriff
restrict 127.127.1.0

restrict 127.0.0.1
restrict 192.168.2.0 mask 255.255.255.0

restrict default notrust nomodify nopeer

##############################
```

Listing 2.1: Beispiel einer ntp.conf-Datei

In der unter *driftfile* angegebenen Datei werden die bereits ermittelten Abweichungen zur Systemzeit gespeichert. Auf der Grundlage dieser Informationen, kann NTP, auch ohne bestehende Verbindung zu einem Zeitserver, die Zeit anpassen. Mit dem Befehl *server* lassen sich die Referenzserver zur Synchronisation bestimmen. NTP stellt hierfür auch Pseudo-IP-Adressen zur Verfügung, um den Zugriff auf spezielle Referenzzeitquellen zu

ermöglichen. Ein Beispiel hierfür ist die *127.127.1.0*, welche den Zugriff auf die eigene Systemzeit ermöglicht. Durch *prefer* lässt sich ein vorzuziehender Zeitserver festlegen, solang dieser nicht zu sehr von den Zeiten der anderen angegebenen abweicht. Der *burst* Befehl ermöglichst eine höhere Synchronität zwischen Server und Client, indem bei einer bestehenden Verbindung zum Zeitserver mehrere, anstatt einem normal üblichen Synchronisierungspaket gesendet werden. Mit dem selben Prinzip bewirkt *iburst* die initiale Synchronisation, wenn noch keine Verbindung besteht. Mit der Kombination der Befehle *fudge* und *stratum* lässt sich weiterhin das Stratum-Level eines Servers manipulieren, um bspw. eine Synchronisation mit gewissen Servern einen Vorrang zu gewähren. Mithilfe von *restrict* wird der Zugriff von anderen Rechnern im Netzwerk oder auch vom localhost selbst beschränkt [ntpa, ntpb]

Kapitel 3

Die Messsoftware

3.1 Konzept zur Latenz- und Datenratenmessung

Das Konzept für die Datenraten- und Latenzmessung besteht darin, zwischen Server und Client in einer Art ”Ping-Pong”-Prinzip abwechselnd Packettrains zu versenden (siehe Abbildung 3.1).

Hierfür beginnt der Client damit, einen Packettrain, mit bekannter Paketanzahl und Größe, an den Server zu senden. Sobald der Server die Pakete erhält, berechnet er für den aktuellen Packettrain Datenrate und Latenz und schickt darauf hin einen eigenen Packettrain zurück an den Client. Der Client verfährt daraufhin auf dieselbe Weise, sodass sich immer nur ein einziger Packettrain zur gleichen Zeit auf der Leitung befindet. Dies ist vergleichbar mit dem Ball in einem ”Ping-Pong”-Spiel, den man hin und her schlägt.

Latenz, Drop- und Datenrate werden in den Paketen gespeichert und mit gesendet, um die Timer berechnen zu können und um zu ermöglichen, dass bei schwankenden Messbedingungen Paketanzahl und Größe angepasst werden können.

Für dieses Konzept sind zuverlässige, verbindungsorientierte Netzwerkprotokolle wie TCP ungeeignet, da durch erneutes senden der Pakete die Messung beeinflusst werden würde. Durch die Verwendung von UDP sind Paketverluste und das Eintreffen der Pakete in falscher Reihenfolge möglich, was für unsere Berechnung von Drop- und Datenrate

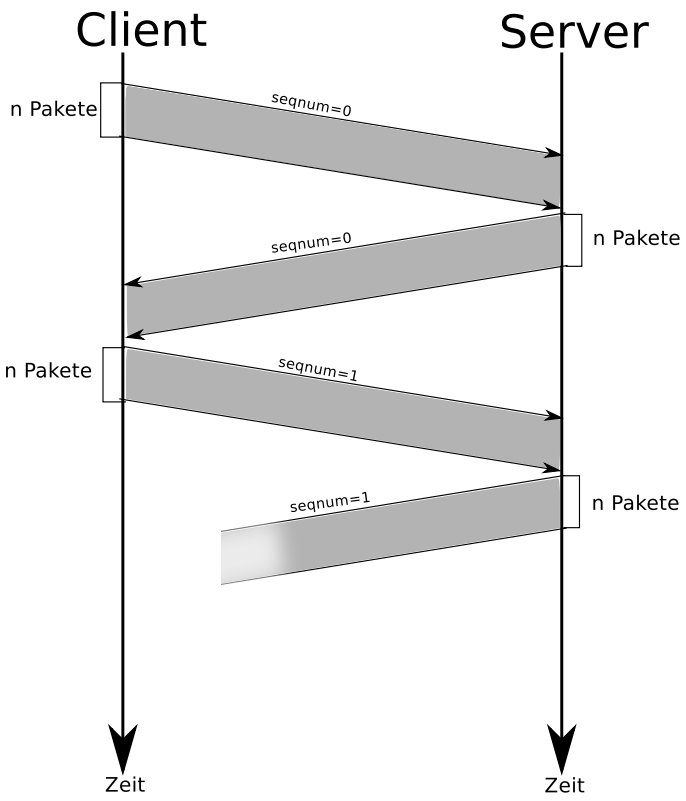

Abbildung 3.1: ”Ping-Pong”-Prinzip

notwendig ist.

3.2 Aufbau eines Messpakets

In einem Messpaket werden diverse Informationen gespeichert, um die Latenz, Drop- und Datenraten berechnen zu können, Timerzeiten zu berechnen und um eine Grundlage für die automatische Anpassung von Größe und Anzahl von Paketen im Packettrain zu schaffen.

Dazu werden Paketgröße s_p und Paketanzahl n im Paket gespeichert, um die oben (in Abschnitt 2.1) angegebenen Berechnungen durchführen zu können. Weiterhin enthält jedes Paket eine Sequenznummer und eine Paketnummer, um verlorene oder aus der Reihe geratene Pakete und Packettrains erkennen zu können. Auch die für den zuvor erhalte-

nen Packettrain berechnete Latenz, Drop- und Datenrate werden mit gesandt, auf deren Grundlage die Timer berechnet werden. Als letzte Information wird noch, aufgeteilt in Sekunden und Nanosekunden, der Sendezeitstempel angehangen, welcher auch für die Berechnungen benötigt wird.

3.3 Logging

Für die Auswertung der Messungen und die spätere Evaluierung der Messsoftware ist es notwendig, die Informationen aus den Messpaketen festzuhalten (siehe Listing 3.1). Hierzu werden die in Abschnitt 3.2 beschrieben Informationen eines jeden Paketes in einer sogenannten Log-Datei gespeichert. Dies geschieht sowohl für gesendete als auch für empfangene Pakete, wobei bei letzteren zusätzlich der Empfangszeitstempel mit in die Datei geschrieben wird. Weiterhin wird zu Beginn einer neuen Messreihe ein Zeitstempel für den Startzeitpunkt der Messung hinzugefügt.

```
#Sun Mar 23 22:05:30 2014
#
#packet_size packet_number seqnum delay datarate droprate SEND_t_sec SEND_t_nsec RECV_t_sec RECV_t_nsec
#
500 10 0 0.010061 1163589 0.100000 0 1395608730 942327949 1395608730 943068828
500 10 0 0.010061 1163589 0.100000 1 1395608730 942362308 1395608730 943494266
500 10 0 0.010061 1163589 0.100000 2 1395608730 942369012 1395608730 943948132
500 10 0 0.010061 1163589 0.100000 3 1395608730 942376275 1395608730 944401998
500 10 0 0.010061 1163589 0.100000 4 1395608730 942383259 1395608730 944856844
500 10 0 0.010061 1163589 0.100000 5 1395608730 942389684 1395608730 945311200
500 10 0 0.010061 1163589 0.100000 6 1395608730 942396388 1395608730 945767517
500 10 0 0.010061 1163589 0.100000 7 1395608730 942403092 1395608730 946221383
500 10 0 0.010061 1163589 0.100000 8 1395608730 942409517 1395608730 946676229
500 10 0 0.010061 1163589 0.100000 9 1395608730 942415942 1395608730 947131566
500 10 1 0.010758 1162017 0.120000 0 1395608730 986970974 1395608730 987708928
```

Listing 3.1: Log-Datei

3.4 Threads

Um diese Konzepte umsetzen zu können, liegt dem Messprogramm folgender Aufbau zugrunde (siehe Abbildung 3.2).

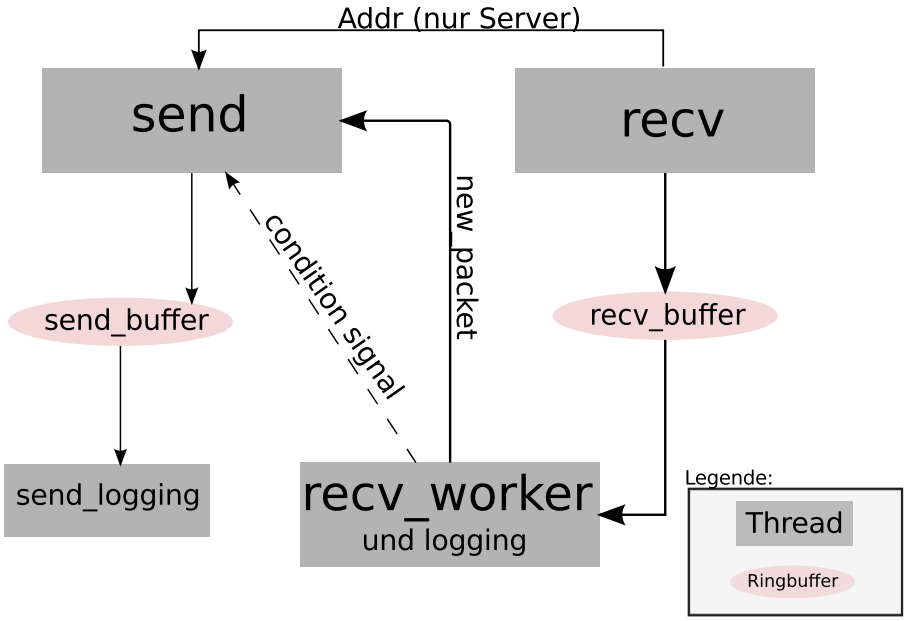

Abbildung 3.2: Thread Zusammenhang

Die einzelnen Elemente für das Senden, Empfangen und Berechnen der Latenz, Drop- und Datenrate sind als Threads implementiert, da die einzelnen Vorgänge gleichzeitig stattfinden. Das Programm besteht aus vier Threads, dem *send*-Thread, zum Senden der Packete, dem *recv*-Thread, zum Empfangen, dem *recv_worker*-Thread, um empfangene Pakete zu loggen und Berechnungen für die Messung auszuführen, sowie dem *send_logging*-Thread, zum Loggen gesendeter Pakete.

Um ein möglichst schnelles Empfangen der Pakete und Anhängen der Empfangszeitstempel zu gewährleisten, wird das eigentliche Empfangen und das Bearbeiten der empfangenen Pakete in getrennten Threads erledigt, im *recv*- und dem *recv_worker*-Thread.

Der *recv*-Thread empfängt durchgehend Pakete, hängt ihnen eine Empfangszeitstempel an und legt sie dann in einen Ringbuffer, den *recv_buffer*. Sobald Pakete im *recv_buffer* vorhanden sind, nimmt der *recv_worker* diese aus dem Buffer und bearbeitet jedes Pa-

ket eines nach dem anderen. Die Aufgaben des *recv_workers* bestehen als erstes darin, die im Paket enthaltenen Informationen zu extrahieren und das Paket zu loggen. Weiterhin betrachtet er die Informationen und berechnet für den aktuellen Packettrain, unter Berücksichtigung von Sonderfällen, wie dem Verlust von Paketen oder auch ganzer Packettrains, die Drop- und Datenrate, sowie die Latenz. Ist der aktuelle Packettrain komplett abgearbeitet, so speichert der *recv_worker* die neu berechneten Daten in einer Speicherstruktur, die die Informationen für zusendende Pakete enthält, dem new_packet und gibt ein Signal an den *send*-Thread. Erhält der *send*-Thread das entsprechende Signal vom *recv_worker*, so nimmt er die in new_packet enthaltenen Daten und generiert einen neuen Packettrain daraus, versieht jedes Paket mit einem Sendezeitstempel und versendet die Pakete. Außerdem, wird der Inhalt des Pakets an einen weiteren Ringbuffer, den *send_buffer*, weitergegeben. Aus diesem entnimmt der *send_logging*-Thread die Pakete wieder, um sie entsprechend eins nach dem anderen zu loggen. Auch *send* und *send_logging* sind jeweils als einzelne Threads realisiert, um sicherzustellen, dass möglichst schnell die Sendezeitstempel gesetzt und die Pakete gesendet werden können.

Sowohl Server als auch Client sind auf diese Weise aufgebaut. Um den Vorgang zu starten, wird auf Clientseite zu Beginn ein Signal an den *send*-Thread gegeben, der daraufhin mit dem Senden des ersten Packettrains beginnt. Kommt dieser auf Serverseite an, wird hier noch IP-Adresse und Portnummer des Clients vom *recv*-Thread an dem *send*-Thread übergeben.

3.5 Timer

Die Messsoftware benötigt zwei Timer, um auf Paketverlust reagieren zu können. Der erste Timer, im weiteren Verlauf als Paket-Timer bezeichnet, wird benötigt, wenn das oder die letzten Pakete eines Packettrains verloren gehen (siehe Abbildung 3.3). Der Timer wird nach dem Empfangen eines jeden Paketes neu gestartet. Bleiben die folgenden Pakete aus, so muss maximal bis zu dem Zeitpunkt gewartet werden, an dem das letzte Paket des aktuellen Packettrains hätte ankommen sollen. Hierzu wird auf Grundlage der für den vorherigen Packetetrain berechneten Datenrate, die Zeit t_{packet} ermittelt, die die noch verbleibenden Pakete benötigen würden, wenn die Datenrate gleich bleibt. Dies

geschieht mit der Formel 3.1. Wobei A für die Datenrate, n für die Anzahl der Paketen im Packettrain, $pnum$ für die Nummer des zuletzt erhaltenen Paketes und s_p für die Paketgröße steht.

$$t_{packet} = \frac{\alpha * (n - pnum) * s_p}{A} \tag{3.1}$$

Das α steht hierbei für einen wählbaren Toleranzparameter. Zu Beginn ist der Packet-Timer auf einen festen Wert gesetzt, und wird, sobald die erforderlichen Daten vorhanden sind, wie oben berechnet.

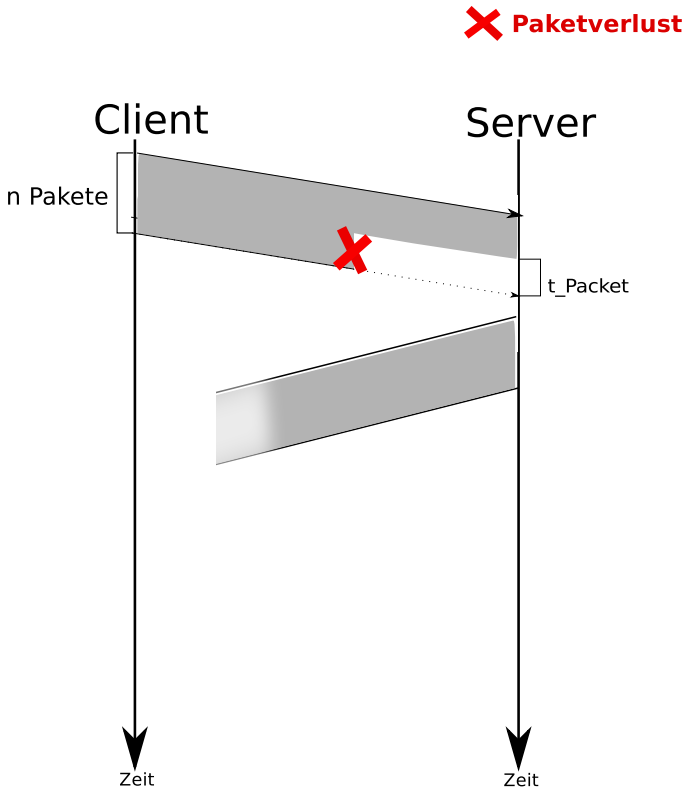

Abbildung 3.3: Paket-Timer-Prinzip

Der zweite Timer, im weiteren als Packettrain-Timer bezeichnet, sorgt dafür, dass nach Ablauf des Timers ein neuer Packettrain gesendet wird, auch wenn ein kompletter Packettrain verloren geht. Der Packettrain-Timer wird zu Beginn eines jeden Sendevorgangs neu gestartet. Es soll die Round Trip Time (RTT) eines kompletten Packettrains gewartet werden, um sicher gehen zu können, dass ein Packettrain verloren gegangen ist. Hierzu messen wir bei jedem Durchgang die Zeit vom Senden eines Packettrains bis zu dem

Zeitpunkt, an dem das letzte Paket des Antwortpackettrains ankommt (siehe Abbildung 3.4). Dieser Wert ist mit einem wählbaren Toleranzfaktor β die Größe unseres Timers, wobei t_{send} den Zeitpunkt darstellt, an dem mit dem Senden begonnen wird und t_{recv} der Zeitpunkt ist, an dem das letzte Paket des Antwortpackettrains ankommt (Gleichung 3.2).

$$t_{train} = \beta * (t_{recv} - t_{send}) \tag{3.2}$$

Tritt Paketverlust am Ende eines Packettrains auf, so würden wir mit dieser Berechnung allerdings die Round Trip Time eines Packettrains ohne Paketverlust unterschätzen. Daher werden nur die RTTs für den Packettrain-Timer genutzt, bei denen das letzte Paket auch tatsächlich angekommen ist. Der Packettrain-Timer ist ebenso zu Beginn auf einem festen Wert gesetzt, welcher dann durch den Berechneten ersetzt wird.

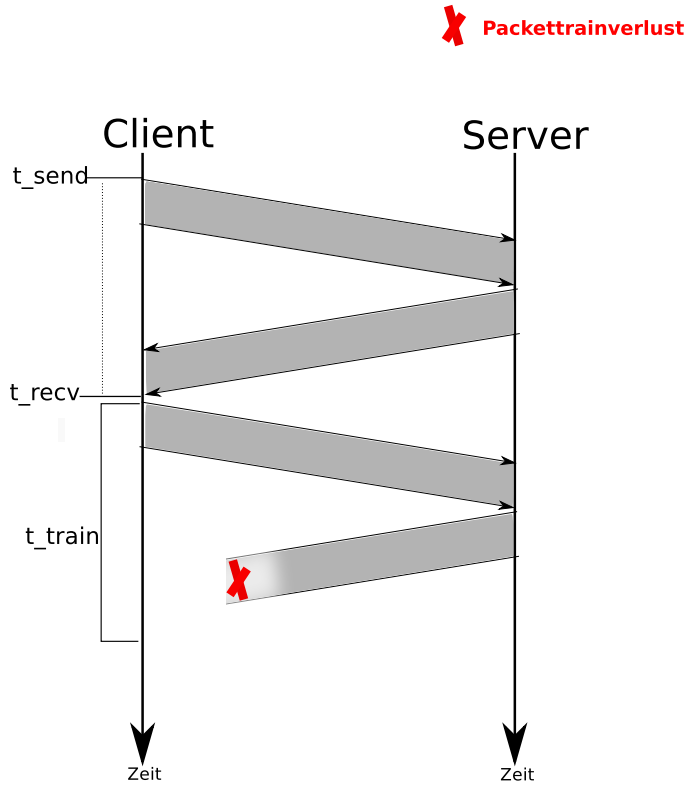

Abbildung 3.4: Packettrain-Timer-Prinzip

Kapitel 4

Evaluation

4.1 Messungen unter Laborbedingungen

Da das Mobilfunknetz diversen Schwankungen unterliegt wurde zur Evaluierung die Messsoftware in einem drahtgebundenen Netzwerk getestet. Das für die Messungen als Server genutzte System hat einen AMD Athlon 64 3000+ (1.8 GHz) Prozessor und 500 MB RAM mit einem VIA Vt6102 Rhine II Fast Ethernet Adapter. Als Client dient ein Acer Aspire 6530 mit AMD Turion X2 Dual Core (2.1 GHz) und 3 GB RAM, sowie einem Qualcomm Atheros AR8172 Fast Ethernet Controller. Als Betriebssystem läuft auf beiden Systeme Lubuntu 13.10 32 Bit. Die beiden Systeme wurden über einen 100 Mbit/s Switch verbunden und die Netzwerkkarten der Systeme mit Hilfe des Programms *ethtool* auf 10 Mbit/s konfiguriert. Weiterhin wurde der Paketverlust des Netzwerks künstlich angepasst, um die realen Verhältnisse im Mobilfunk zu simulieren. Daraufhin wurden Messungen mit verschiedenen Pakettraingrößen durchgeführt und überprüft, ob die Messsoftware die zu erwartenden Daten- und Dropraten berechnet.

4.1.1 Anpassung von Paketverlust

Um die Funktionalität der Messsoftware zu überprüfen war es notwendig, das Netzwerk, in dem gemessen wird, anzupassen, um den Verlust von Paketen zu simulieren. Zur An-

passung des Paketverlusts im Netzwerk wurde *iptables* genutzt. Das Programm *iptables* ermöglicht dem Nutzer die Bearbeitung Firewalltabellen des Linuxkernels. Diese Tabellen enthalten Ketten von Regeln, die das weiterleiten von Netzwerkpaketen beeinflussen. Somit ist es uns möglich eingehenden Pakete nicht weiterzuleiten und somit Paketverlust zu simulieren. *iptables* bietet zudem die Möglichkeit zufällig Pakete mit einer bestimmbaren Wahrscheinlichkeit nicht weiterzuleiten, was uns eine Grundlage zum Vergleich für die Messsoftware bietet.

```
sudo iptables -A INPUT -m statistic -mode random -probability
0.03 -j DROP
```

Mit *-A INPUT* wird angegeben, dass eine neue Regel für die eingehenden Pakete hinzugefügt werden soll. Die *iptables* Erweiterung *statistic* erlaubt es uns, Pakete nach statistischen Bedingungen auszuwählen. In diesem Beispiel geschieht das zufällig, mit einer Wahrscheinlichkeit von 0.03, angegeben durch die Befehle *– mode random –probability 0.03 . -j DROP* gibt an, dass die Pakete, auf die diese Regel zutrifft, verworfen werden sollen. Der angegebene *iptables* Befehl bewirkt also, dass ein eingehendes Paket, mit einer Wahrscheinlichkeit von 0,03 verworfen wird. [iptb, ipta, dro]

4.1.2 Überprüfung mit iperf-Messungen

Um die tatsächliche Daten- und Droprate des Netzwerks feststellen zu können, wurde das Programm *iperf* benutzt. *iperf* ist ein auf Client und Server basierendes Messprogramm, welches es ermöglicht Datenrate, Jitter und Paketverlust zwischen zwei Rechnern in einem Netzwerk zu messen. Es kann sowohl mit TCP als auch UDP genutzt werden. Somit ist *iperf* dafür geeignet, zu überprüfen, ob unsere Verbindung die von uns erwartete verfügbare Datenrate und Droprate besitzt.

```
$ iperf -s -u -b 10M
```

Listing 4.1: Iperf Server

Für die Evaluierung lassen wir *iperf* im UDP-Modus laufen (mittels des Parameters *-u*) und erhöhen die Bandbreite mit der *iperf* sendet mittels des Paramters *-b 10M* auf

10 Mbit pro Sekunde. Wir nutzen *iperf* mit den in Listing 4.1 gezeigten Parametern, für den Server und in Listing 4.2 für den Client. Da der default Wert für den UDP Modus bei 1Mbit/s liegt, erhöhen wir dies auf mindestens 10 Mbit/s, um die 10 Mbit-Verbindung voll auslasten zu können. [ipeb, ipea]

```
$ iperf -c 10.0.0.1 -u -b 10M
```

Listing 4.2: Iperf Client

4.2 NTP-Konfiguration

Zur Zeitsynchronisation zwischen Client und Server der Messsoftware wurde das in Abschnitt 2.2 beschriebene Network Time Protokoll (NTP) genutzt. Hierzu wurde eines der beiden Systeme als Zeitserver, das andere als normaler NTP-Client eingerichtet.

```
### Server:/etc/ntp.conf ######

# Abweichung
driftfile /var/lib/ntp/ntp.drift
#Server
server 127.127.1.0 iburst burst prefer
fudge 127.127.1.0 stratum 1
# Zugriff von NTP-Server
restrict 127.127.1.0
#Zugriff gestatten
restrict 127.0.0.1
restrict 10.0.0.0 mask 255.255.255.0
#default verweigern
restrict default notrust nomodify nopeer
```

Listing 4.3: NTP-Konfiguration Server

Auf Serverseite wurde als einzige Zeitquelle , mit Hilfe der Pseudo-IP-Adresse 127.127.1.0, die lokale Systemzeit genutzt, da nur die beiden Systeme zueinander synchron sein müssen. Eine Synchronisation mit einer externen Zeitquelle ist also in unserem Fall nicht

notwendig. Durch *iburst* und das Festsetzen des Stratum Levels auf 1, wird die anfängliche Synchronisation der Systeme beschleunigt. Mit Hilfe von *burst*, wird die Zeit weiterhin so synchron wie möglich gehalten, da im drahtgebundenen Netz, schon kleine Abweichungen zu signifikanten Änderungen bei der Latenz führen können.

```
### Client:/etc/ntp.conf ######
# Abweichungen
driftfile /var/lib/ntp/ntp.drift
# NTP-Server
server 10.0.0.1 iburst burst prefer
# Zugriff
restrict 10.0.0.1
restrict 127.0.0.1
# default: verwehren
restrict default notrust nomodify nopeer
```

Listing 4.4: NTP-Konfiguration Client

Auch auf Clientseite wurde nur ein Zeitserver, der von uns eingerichtete NTP-Server, genutzt. Weiterhin wurden die selben Konfigurations-Kommandos wie beim NTP-Server genutzt, um die oben bereits genannten Eigenschaften zu ermöglichen.

4.3 Probemessungen für Datenrate

Um die Genauigkeit der Datenratenbestimmung der Messsoftware zu überprüfen, wurden Messungen unter Laborbedingungen auf einer 10 Mbit Ethernet-Verbindung durchgeführt. Hierzu wurden die Netzwerkkarten der Systeme konfiguriert, sodass die Übertragungsgeschwindigkeit 10 Mbit/s beträgt. Dazu wurde das Programm *ethtool* mit dem im folgenden gezeigten Befehl verwendet [eth].

```
ethtool -s eth0 speed 10 duplex full autoneg off
```

Somit ergab sich für diese Verbindung eine maximale verfügbare Datenrate von 1280 *kByte/s*

(siehe Gleichung 4.1).

$$\frac{10\,Mbit/s}{8} * 1024 = 1280\,kBytes/s \tag{4.1}$$

Hier und und in den folgenden Messreihen ist Datenrate auf Ethernetebene angegeben.

Für die Messungen wurden die Parameter der Messsoftware variiert.

Paketanzahl	100, 250, 500, 750, 1000
Paketgröße	500, 750, 1000,1250,1500

Tabelle 4.1: Variation der Messparameter

Es wurden Paketgröße und Paketanzahl in einem Packettrain entsprechend der Werte in Tabelle 4.1 kombiniert. Also obere Grenze für die Paketgröße wurde mit 1500 Bytes die Maximum Transmission Unit (MTU) einer Ethernet-Verbindung gewählt. Pakete, die größer als die MTU sind, werden von der Netzwerkkarte fragmentiert gesendet, also auf einzelne kleinere Pakete aufgeteilt. Bei der Berechnung von Daten- und Droprate unseres Messverfahrens würde dies zu Fehlern führen. Weiterhin wurde für die Packettraingröße als Maximum die Kombination 1000x1250 Bytes gewählt, da dies bereits sehr nahe an die maximal verfügbare Datenrate von 1280 kBytes/s für Ethernet heran reicht. Hier und im weiteren definieren wir die Pakettraingröße $size_{pt}$ wie folgt,

$$size_{pt} = s_p * n \tag{4.2}$$

wobei n die Anzahl der Pakete und s_p ihre Größe darstellen.

In Abbildung 4.1 wurden für die Messungen beispielhaft die gemessenen Datenraten von je 100 Packettrains aufgetragen. Hierbei wurde die Anzahl der Pakete in einem Packettrain konstant gehalten und die Paketgröße variiert. Abbildung 4.3 besitzt die gleichen Achsenbeschriftungen, allerdings wurde hier ein fester Wert für die Paketgröße gewählt und die Anzahl der Pakete variiert. Zur Veranschaulichung der Messausreißer sind in Abbildung 4.2 und 4.4, die maximalen und minimalen Werte, sowie die Mittelwerte der selben Daten als Fehlerbalken angegeben.

Es lässt sich erkennen, dass die gemessenen Datenraten, trotz variierten Paketgrößen und Anzahl, im Mittel einen konstanten Wert annehmen. Jedoch ist festzustellen, dass vor

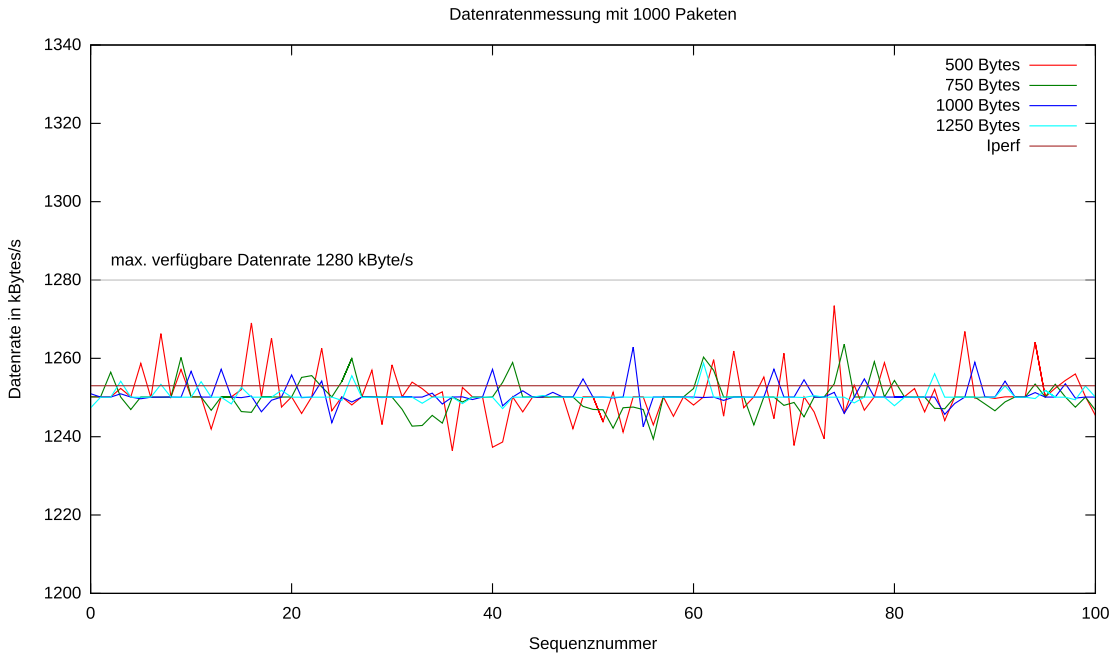

Abbildung 4.1: Datenraten mit fester Paketgröße

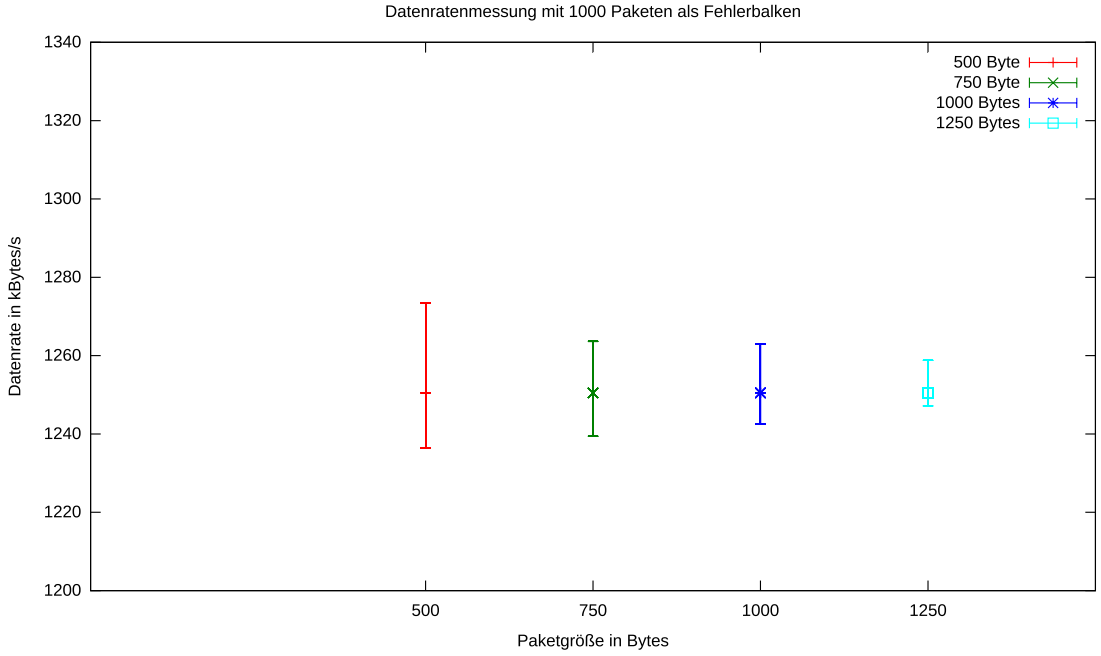

Abbildung 4.2: Fehlerbalken mit fester Paketgröße

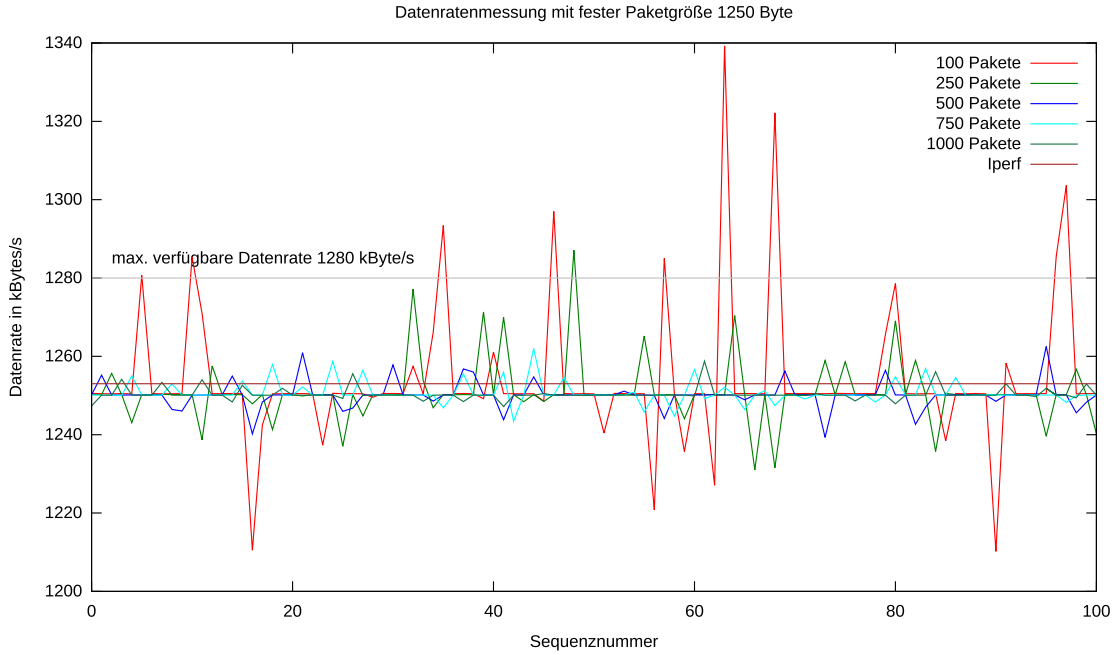

Abbildung 4.3: Datenraten mit fester Paketanzahl

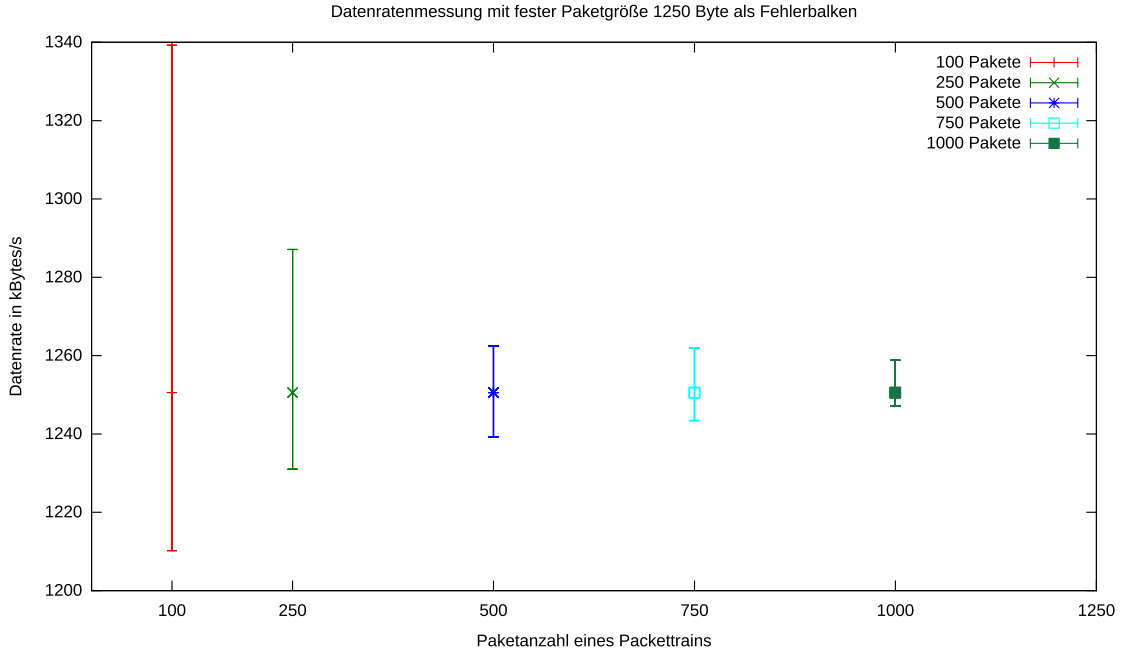

Abbildung 4.4: Fehlerbalken mit fester Paketanzahl

allem bei kleinen Packettrains starke Messausreißer auftreten. Diese Messausreißer werden mit zunehmender Packettraingröße kleiner. Um diesen Effekt deutlicher darzustellen sind in Abbildung 4.5 und 4.6 die selben Messwerte als kumulative Verteilungsfunktion aufgetragen. Auch hier sind die größeren Messausreißer bei kleinen Packettrains zu erkennen. Dies ist ein durchaus zu erwartener Effekt, da sich kleine Schwankungen der Zeitstempel für die ersten und letzten Pakete eines Packettrains deutlich stärker auf die Berechung der Datenrate auswirken können, wenn die Packettraingröße gering ist.

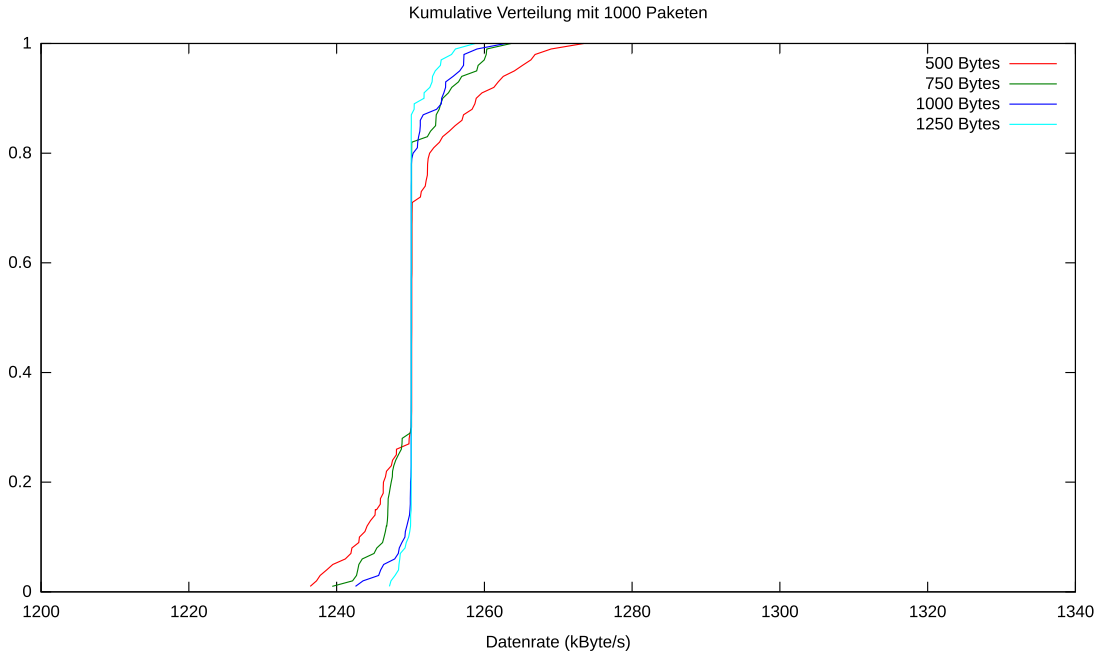

Abbildung 4.5: kumulative Verteilung für Mesungen mit fester Paketgröße

Weiterhin sind in Abbildung 4.7 für eine feste Packettraingröße von 150000 Byte die gemessenen Datenraten für die Kombinationen 100x1500 Byte, 200x750 Byte, 300x500 Byte, sowie 400x250 Byte. Hier fällt auf, dass trotz gleicher Packettraingröße, die Kombinationen mit großen Paketgrößen kleinere Messausreißer aufweisen als Packettrains mit kleinen Paketen.

Auf Grund dieser Ergebnisse, wurde für weitere Messungen eine hohe Anzahl von Paketen ausgewählt. Außerdem verspricht eine hohe Paketgröße, Messergebnisse mit weniger Messausreißern. Für die weiteren Messungen wurde daher die Kombination 1000x1250 Byte für die Packettraingröße ausgewählt, damit wird, wenn keine Timer ausgelöst werden, je-

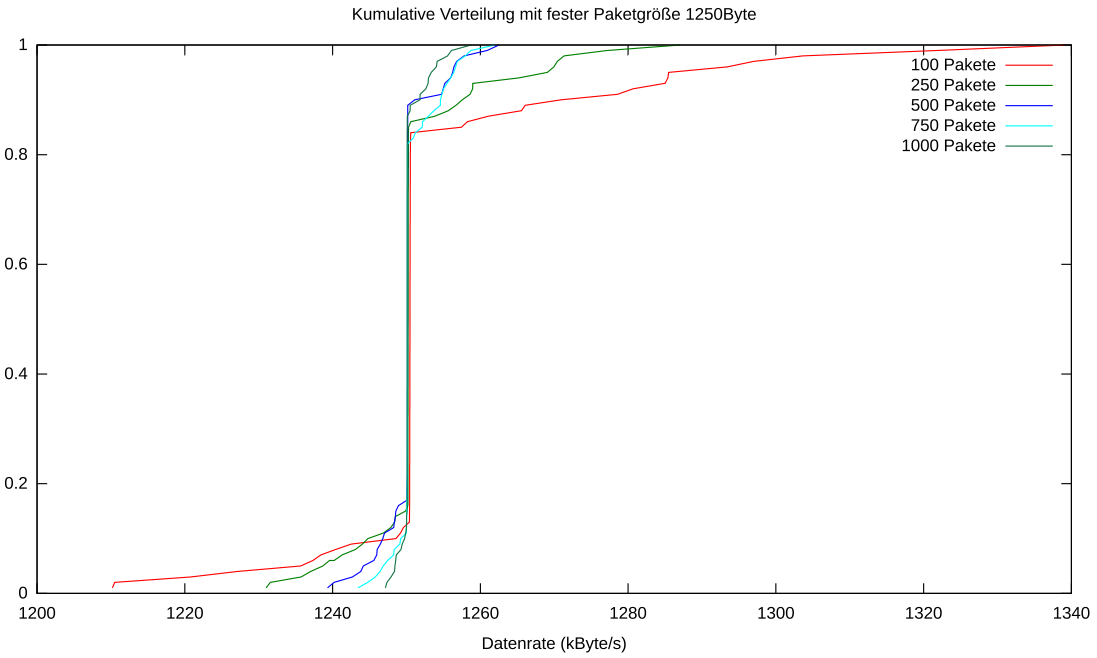

Abbildung 4.6: kumulative Verteilung für Mesungen mit fester Paketanzahl

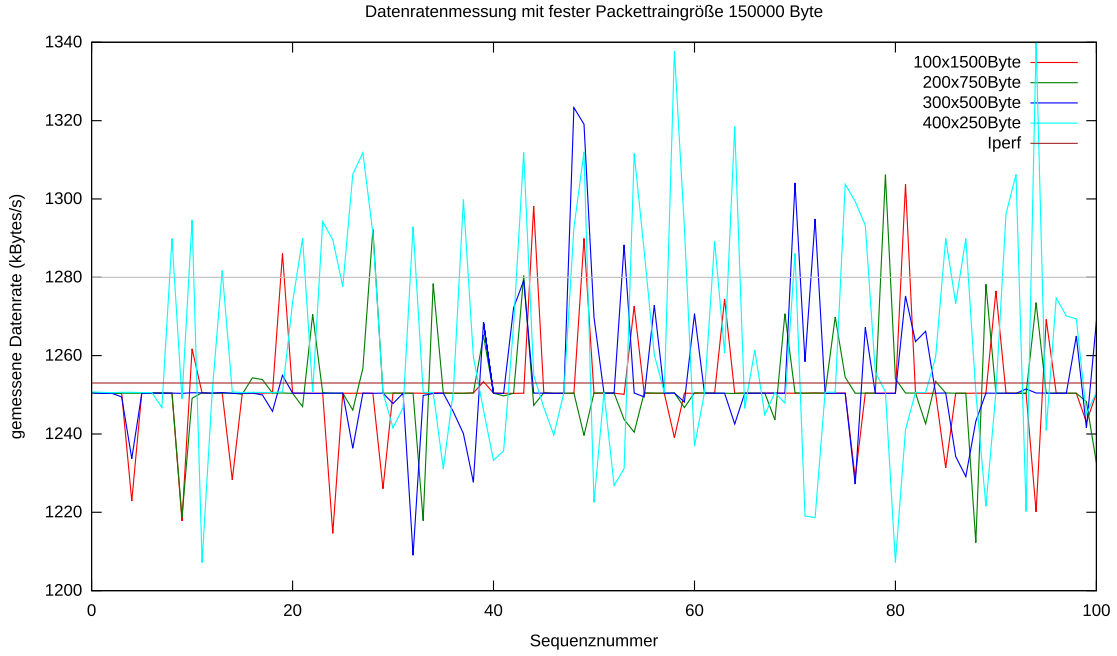

Abbildung 4.7: Datenrate bei gleichbleibender Packettraingröße

de Sekunde ein Messwert für eine Messrichtung generiert. Somit werden Messwerte auf Client- und Serverseite jeweils mit einer Frequenz von 0.5 Hz aufgenommen.

Um mögliche Abweichungen bei Betrieb der Messsoftware über einen längeren Zeitdauer zu erkennen, wurde mit mit einem Pakettrain von 1000x1250 Byte eine Messung über 600 Sekunden genommen. (siehe Abbildung 4.8). Außerdem wurde hier die Messungen für beide Messrichtungen aufgetragen.

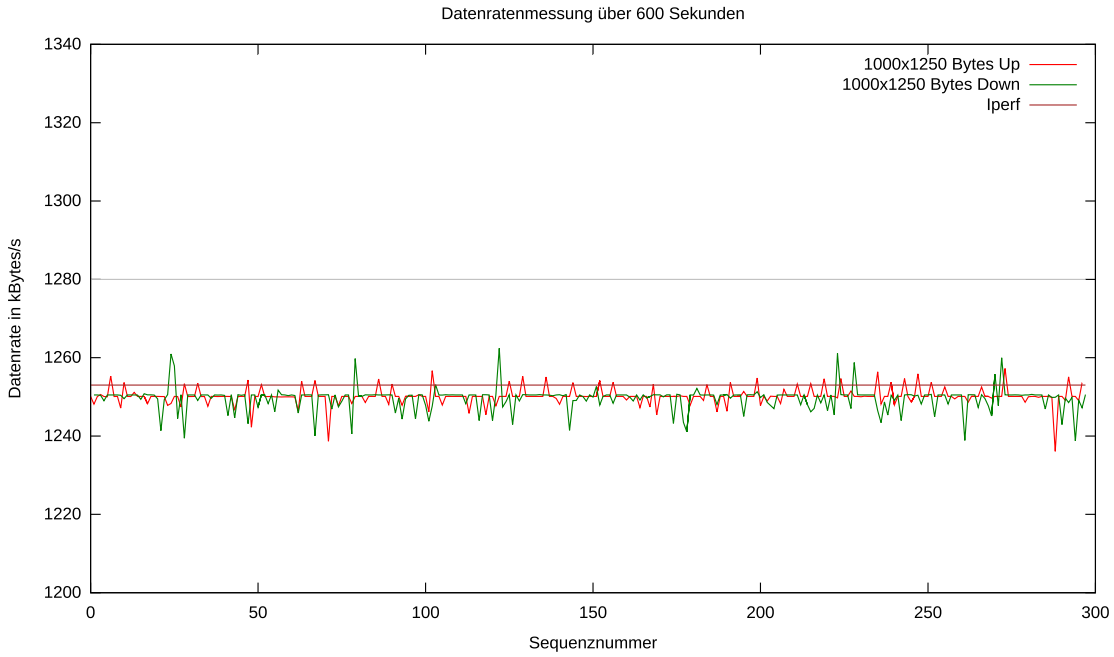

Abbildung 4.8: Datenratenmessung über einen Zeitraum von 600s

Die Messungen und deren Ausreißer bleiben hierbei für beide Messrichtungen im selben Rahmen, wie bei den zuvor durchgeführten, kürzeren Messreihen (vergleiche Abbildung 4.2).

4.3.1 Probemessungen mit angepasstem Paketverlust

Um das Verhalten der Messsoftware bei Paketverlust beobachten zu können, wurden mit Hilfe von *iptables* (siehe Abschnitt 4.1.1) verschiedene Dropraten simuliert. Um sinnvoll

vergleichen zu können, wurden Messungen mit festen Werten für Paketgröße und Paketanzahl eines Packettrains durchgeführt und der Paketverlust varriert. Es wird wie zuvor festgelegt, mit 1000 Paketen zu je 1250 kBytes gemessen. In Abbildung 4.9 wurden

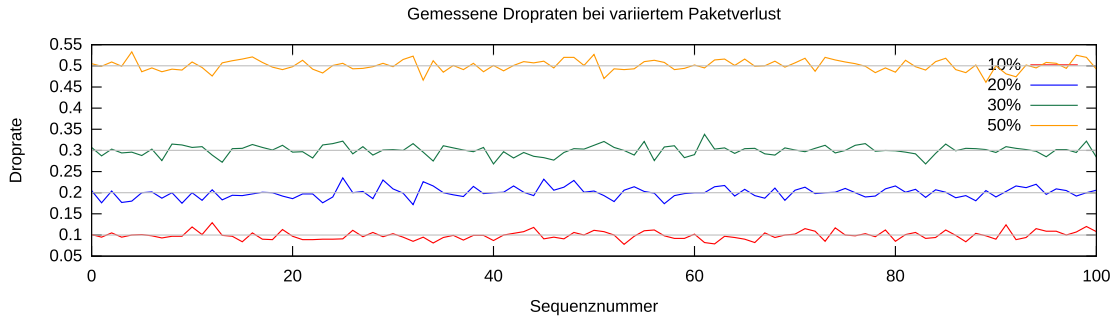

Abbildung 4.9: Dropmessung bei 1000x1250 Bytes

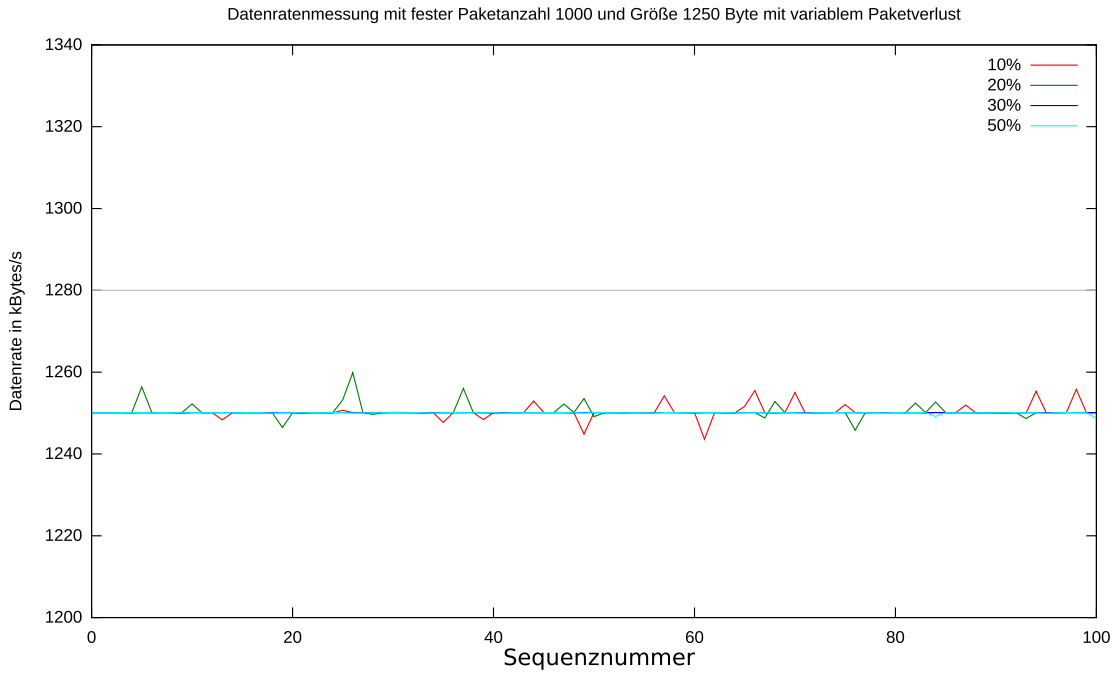

Abbildung 4.10: Datenratenmessung bei 1000x1250 Bytes

die gemessenen Dropraten von wie zuvor je 100 Packettrains aufgetragen. Die einzelnen Graphen zeigen die Messungen bei erzeugtem Paketverlust von 10%, 20% 30%, sowie 50%. Als Beispiel dient uns hier eine Messung mit 1000 Paketen zu je 1250 Byte. Abbildung 4.10 zeigt die dazugehörigen Datenraten. Die Schwankungen in der Droprate

können darauf zurückgeführt werden, dass *iptables* für jedes Paket per Zufall bestimmt, ob es verworfen wird oder weitergeleitet wird. Somit schwankt die Prozentzahl der insgesamt verworfenen Pakete pro Packettrain leicht um die angelegte Droprate. Allerdings liefern, die Mittelwerte der Dropratenmessungen Zahlen, welche den anlegten Dropraten, womit davon ausgegangen werden kann, dass die Droprate korrekt berechnet wird.

4.4 Bestimmung der Timer-Variablen

Um die Toleranzparameter α und β für die Timer zu bestimmen (siehe Abschnitt 3.5) wurden weitere Messungen mit 1000x1250 Byte Packettrains durchgeführt.

Für Messungen mit verschiedenen α-Werten für den Packet-Timer wurden bei den Messungen die Dropraten varriert. Um sicher zu stellen, dass der Packettrain-Timer keinen Einfluss auf die Messungen für verschiedene α-Werte nimmt, wurde er auf einen für die Messbedingungen hohen Wert von $\beta = 5$ gesetzt. Da in diesem Fall kaum große Latenzschwankungen auftreten, genügt ein 5-faches der RTT eines Packettrains bei weitem, um verfrühte Timeouts vermeiden zu können. Außerdem ist bei den getesteten Dropraten kein Packettrainverlust zu erwarten, somit finden auch keine gewollten Timeouts für die Packettrains statt, und nur Timeouts, welche durch den Paket-Timer ausgelöst werden fließen in die Messung ein.

In Abbildung 4.11 betrachten wir die Anzahl der aufgetretenen verfrühten Timeouts durch den Packet-Timer bei einer festen Messdauer von 600 Sekunden. Hierfür wurde für verschiedene Dropraten der Wert α des Paket-Timers varriert. Hierbei fällt auf, dass bei hohen Dropraten gar keine verfrühten Timeouts auftreten. Erst bei Droprate 10% und kleiner werden Timeouts verfrüht ausgelöst. Die Anzahl der verfrühten Timeouts hat zwischen 7,5% und 2,5% ihren Höhepunkt und nimmt dann wieder ab.

Abbildung 4.12 zeigt für die selben Messwerte die Anzahl der empfangen Packettrains in der festen Messedauer von 600 Sekunden für verschiedene Dropraten bei variierten α im Vergleich zum Idealfall mit 0% Droprate. Wie zu erwarten war, sinkt die Anzahl der Packettrains im festen Zeitintervall mit Anstieg der Droprate. Dies liegt daran, dass bei

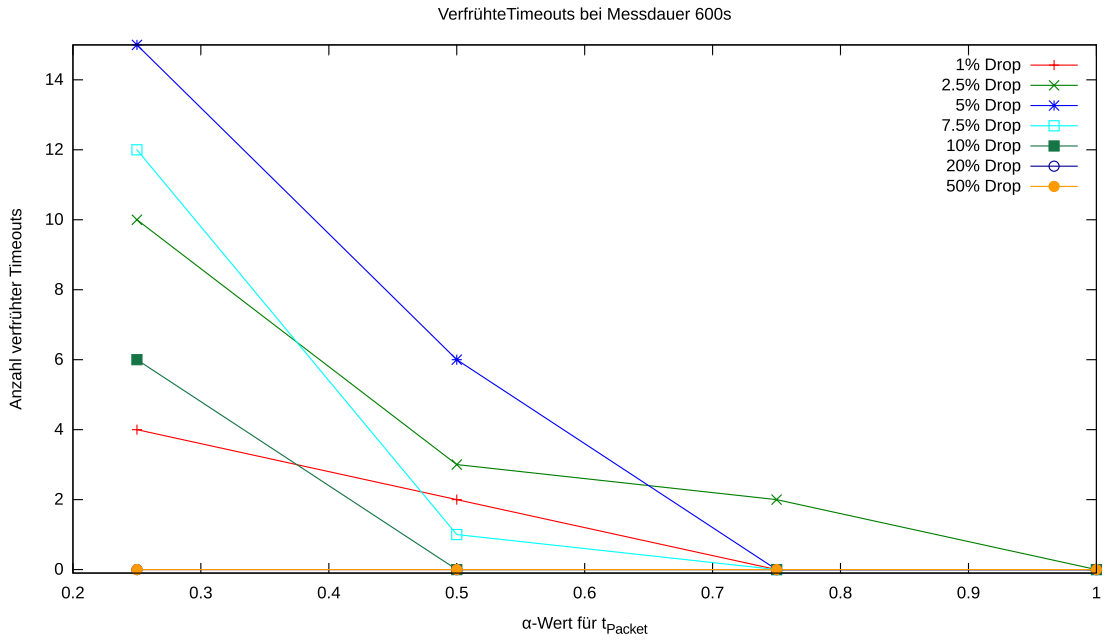

Abbildung 4.11: Verfrühte Paket-Timeouts bei fester Messdauer

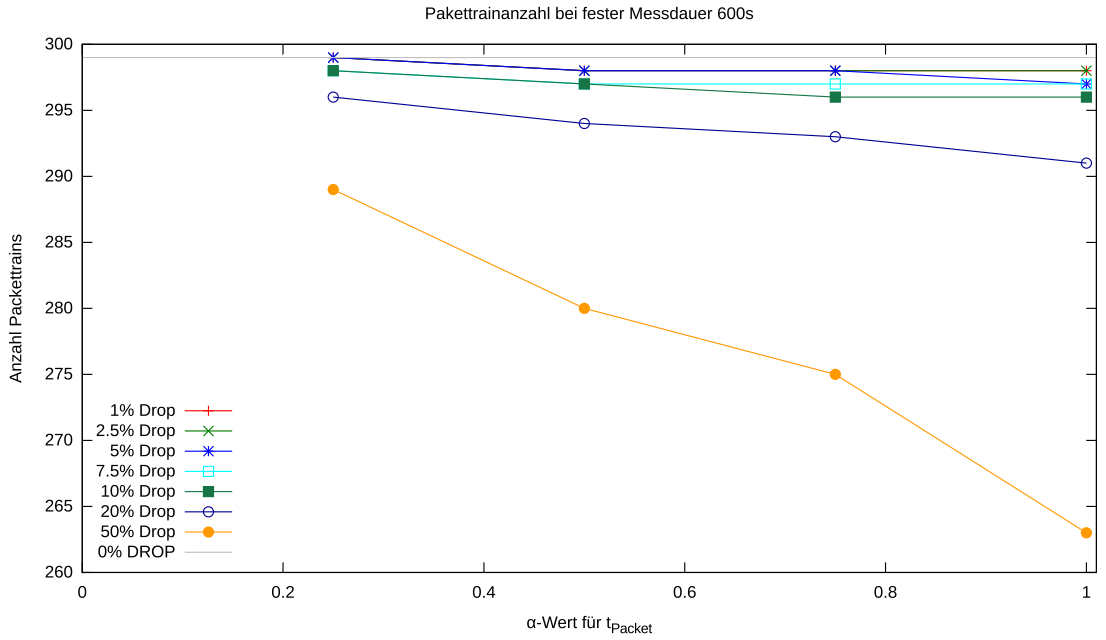

Abbildung 4.12: Empfangene Packettrains bei fester Messdauer

höherem Paketverlust auch öfters Timeouts ausgelöst werden, da Pakete am Ende eines Pakettrains verloren gehen. Dies wirkt sich stärker auf die Messdauer aus, da immer wenn ein Packettrain nicht komplett ankommt, zusätzlich gewartet werden muss und somit weniger Packettrains in einem festen Zeitintervall gesendet werden können.

Für die weiteren Messungen wurde der α-Wert auf 1 gesetzt, zwar ist vor allem bei hohen Dropraten hierbei mit deutlich weniger Pakettrains pro Messung zu rechnen, jedoch zeigen die Messungen bei kleineren α-Werten eine Chance auf verfrühte Timeouts. Für Verbindungen bei der höhere Dropraten zu erwarten sind, wäre es möglich eine Wert kleiner als 1 für α zu wählen, um die Anzahl der Packettrains pro Zeitintervall steigern zu können.

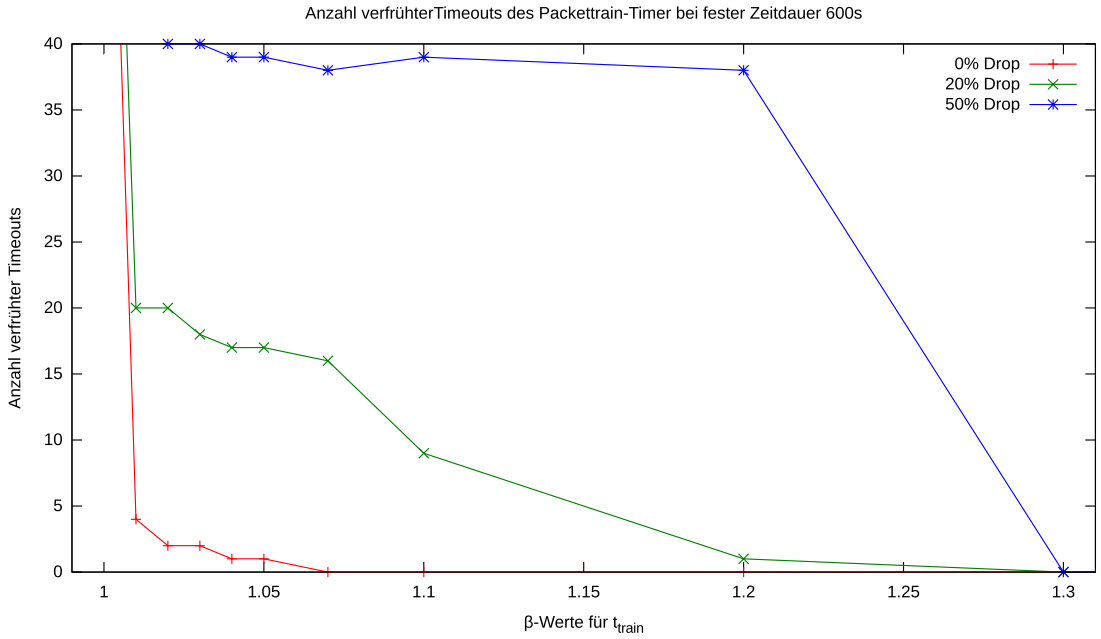

Abbildung 4.13: verfrühte Packettrain-Timeouts bei fester Messdauer

In Abbildung 4.13 ist für verschiedene β-Werte die Anzahl der verfrühten Timeouts aufgetragen. Ist $\beta = 1$ gewählt, so ist der Timeout genau auf die Zeitdauer gesetzt, welche der zuvor gesendete Packettrain benötigt hat. Es ist festzustellen, dass man für β den Wert 1.1 als untere Grenze wählen sollte, da sonst trotz einer Droprate von 0%, also bei keinerlei Verlust von Packettrains, Packettrain-Timeouts auftreten. Für $\beta = 1$ steigt die Anzahl der verfrühten Timeouts sogar auf 90 von 300 Packettrains. Dies ist darauf

zurückzuführen, dass in den Round Trip Times der einzelnen Pakete leichte Schwankungen auftreten. Für höhere Dropraten, sollte auch ein höherer β-Wert gewählt werden. Für eine Droprate von 20% ist erst ab $\beta = 1.2$ ein akzeptabler Wert von nur einem einzelnen verfrühten Timeout aufgetreten. Gar keine verfrühten Timeouts kamen bei den Messungen für 20% und 50% erst bei $\beta = 1.3$ vor.

Kapitel 5

Zusammenfassung und Ausblick

Im Rahmen dieser Bachelorarbeit wurde eine aus Server und Client bestehende Messsoftware für Latenz, Daten- und Dropraten implementiert, welche auf einem ”Ping-Pong”-Prinzip für die Messpackettrains basiert. Weiterhin wurde durch Anpassen von Daten- und Droprate der Verbindung unter Laborbedingungen untersucht, ob die gemessen Daten- sowie Dropraten, den realen Bedingungen der Verbindung entsprechen. Es wurde verglichen, inwiefern die Größe und Anzahl der Pakete in einem Packettrain Auswirkungen, auf die Genauigkeit der Messung hat. Außerdem wurden die Timer, die im Messprogramm bei Verlust von Paketen und Packettrains für einen Timeout sorgen, gewählt und verschiedene Parameter für die Toleranz getestet, um Grenzen für diese Werte zu finden.

Im Laufe der Bachelorarbeit taten sich noch weiterführende Fragen und Möglichkeiten der Verbesserung, welche nicht mehr durchgeführt werden konnten, auf. Diese Möglichkeiten sind im folgenden aufgelistet:

- Automatische Anpassung von Paketgröße und Anzahl, sowie Timern, beisich ändernden Bedingungen der Verbindung (bspw. schwankende Datenraten oder sich ändernden Mobilfunkstandard).

- Die Berechnung der Timer über mehrere Messungen bestimmen bzw. schätzen, anstatt nur Werte des letzten Durchgangs zu betrachten.

- Auswirkungen von variierenden Latenzen auf die Messwerte und Timer.
- Genauere Bestimmung der Timerparameter, im Speziellen unter realen Mobilfunkbedingungen.
- Probemessungen für Latenz, Daten- und Droprate im realen Mobilfunk.
- Messungen unter Bewegung im Mobilfunknetz.

Literaturverzeichnis

[Chr13] CHRISTIAN SIMON LANGE: *Untersuchung der Auswirkungen paralleler Datenratenmessungen in einer Mobilfunkzelle* , Heinrich-Heine Universität Düsseldorf, Masterarbeit, 2013

[dro] *Dropping Packets in Ubuntu Linux using tc and iptables*. http://sandilands.info/sgordon/dropping-packets-in-ubuntu-linux-using-tc-and-iptables.

[eth] *Ethtool Man Page*. http://www.clearfoundation.com/docs/man/index.php?s=8&n=ethtool.

[GKMG14] GOEBEL, Norbert; KOEGEL, Markus; MAUVE, Martin; GRAFFI, Kalman: Trace-based Simulation of C2X-Communication using Cellular Networks. In: *11th Annual Conference on Wireless On-demand Network Systems and Services (WONS 2014)*. Obergurgl, Austria, apr 2014.

[ipea] *Iperf*. http://staff.science.uva.nl/~jblom/gigaport/tools/man/iperf.html.

[ipeb] *Iperf - The TCP/UDP Bandwidth Measurement Tool*. http://iperf.fr/#service.

[ipta] *Iptables-Extension Man Page*. http://ipset.netfilter.org/iptables-extensions.man.html.

[iptb] *Iptables Man Page)*. `http://ipset.netfilter.org/iptables.man.html`.

[Mil85] MILLS, D.L.: *Network Time Protocol (NTP)*. RFC 958. `http://www.ietf.org/rfc/rfc958.txt`. Version: September 1985 (Request for Comments). Obsoleted by RFCs 1059, 1119, 1305

[Mil88] MILLS, D.L.: *Network Time Protocol (version 1) specification and implementation*. RFC 1059. `http://www.ietf.org/rfc/rfc1059.txt`. Version: Juli 1988 (Request for Comments). Obsoleted by RFCs 1119, 1305

[Mil92] MILLS, D.: *Network Time Protocol (Version 3) Specification, Implementation and Analysis*. RFC 1305 (Draft Standard). `http://www.ietf.org/rfc/rfc1305.txt`. Version: März 1992 (Request for Comments). Obsoleted by RFC 5905

[MMBK10] MILLS, D.; MARTIN, J.; BURBANK, J.; KASCH, W.: *Network Time Protocol Version 4: Protocol and Algorithms Specification*. RFC 5905 (Proposed Standard). `http://www.ietf.org/rfc/rfc5905.txt`. Version: Juni 2010 (Request for Comments).

[Nor10] NORBERT GOEBEL: *Trace-basierte Simulation von Mobilfunkcharakteristiken für die Fahrzeug-zu-Fahrzeug Kommunikation* , Heinrich-Heine Universität Düsseldorf, Masterarbeit, 2010

[ntpa] *Manual Reference Pages - NTP.CONF*. `http://www.gsp.com/cgi-bin/man.cgi?topic=ntp.conf`.

[ntpb] *Zeitsynchronisation mit NTP (Client/Server)*. `http://www.linux-fuer-alle.de/doc_show.php?docid=7`.

[Seb12] SEBASTIAN WILKEN: *Verfahren zur Datenratenmessung in Mobilfunknetzwerken* , Heinrich-Heine Universität Düsseldorf, Masterarbeit, 2012

Ehrenwörtliche Erklärung

Hiermit versichere ich, die vorliegende Bachelorarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt zu haben. Alle Stellen, die aus den Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Düsseldorf, 10.April 2014 Niklas Först

Diese CD enthält:

- eine *pdf*-Version der vorliegenden Bachelorarbeit
- die LaTeX- und Grafik-Quelldateien der vorliegenden Bachelorarbeit samt aller verwendeten Skripte.
- die Quelldateien der Messsoftware
- Die in der Arbeit verwendeten Messreihen
- die Websites der verwendeten Internetquellen