Leopold-Franzens-Universität
Innsbruck

Institut für Informatik

Analyse des Netzwerkverhaltens von Computerspielen und Echtzeit-Multimedia-Internetanwendungen detaillierte Analyse von Fifa Football, Jedi Knight II und iVisit

Bakkalaureatsarbeit

eingereicht bei Dr.-Ing. Michael Welzl

Autor: Roland Wallnöfer

Innsbruck, 28.02.2005

Inhaltsverzeichnis

1 Allgemeines

1.1 Einleitung

Diese Arbeit baut zum Teil auf meine erste Bakkalaureatsarbeit [1] auf, welche sich mit der Analyse des Netzwerkverhaltens von Videoconferencing Tools befasste. Weiters sollte das Netzwerkverhalten von zwei Computerspielen (Fifa Football 2004 und Jedi Knight II) genauer untersucht werden. Das Netzwerkverhalten dieser beiden Computerspiele wurde bereits in einer Bakkalaureatsarbeit [2] von zwei Studienkollegen untersucht. Auffallend bei dem Videoconferencing Tool und den zwei Computerspielen war in den vorangegangenen Analysen, dass diese bei den verschiedenen Szenarien eine Art Staukontrolle aufweisen und auf verschiedene Verkehrssituationen reagieren.

In dieser Arbeit sollen das Videoconferencing Tool (iVisit) und die Computerspiele anhand eines Netzwerkemulators (Nistnet), der auf einem Router installiert wird, durch Emulation verschiedener „delay-Zeiten“ und „drop-Raten“ genauer untersucht werden.

1.2 Aufgabenstellung

Ziel dieser Arbeit ist es, mit Hilfe des Emulators NistNet die Computerspiele (Fifa Football 2004 und Jedi Knight II) und die Videoconferencing Anwendung iVisit durch Emulation verschiedener „delay-Zeiten“ und „drop-Raten“ durch Aufzeichnen des Netzwerkverkehrs zu evaluieren und die Ergebnisse auszuwerten. Der Netzwerkemulator NistNet soll auf einen dazwischen geschalteten Router installiert werden.
Durch die Ergebnisse kann das Verhalten der jeweiligen Anwendung im realen Netzwerk in Bezug auf Durchsatz, Pakete pro Sekunde und der durchschnittlichen Paketlänge pro Sekunde bestimmt werden. Um eine Übersicht der einzelnen Anwendungen im gesamten Testbereich zu bekommen, sollen die oben genannten Kennwerte (Durchsatz, Pakete pro Sekunde und durchschnittliche Paketlänge pro Sekunde) jeweils für jedes Testszenario aufsummiert und in einem Diagramm dargestellt werden. Dadurch sollen für jede Anwendung jeweils für die verschiedenen „delay-Zeiten“ und „drop-Raten“ die Diagramme

- sum throughput
- sum number of packets
- average packet length

erstellt werden. Wobei zur Darstellung der durchschnittlichen Paketlänge der aufsummierte Wert noch jeweils durch 120 (jeweilige Versuchsdauer 120s) dividiert wird.

2 Theoretischer Hintergrund

2.1 Was ist Emulation?

Als Emulation wird in der Computertechnik das funktionelle Nachbilden eines Systems durch ein anderes bezeichnet. Das nachbildende System erhält die gleichen Daten, führt die gleichen Programme aus und erzielt die gleichen Ergebnisse wie das nachgebildete System. Ein Emulator ist ein System, das ein anderes emuliert. Zu unterscheiden sind Hardware- und Software-Emulatoren.
Ein Hardware-Emulator ist ein elektronisches Gerät, das z.B.: einen Mikroprozessor funktionell, elektrisch und mechanisch (z.B.: Pins) nachbilden kann. Ein Software-Emulator ist eine virtuelle Maschine. Im Vordergrund steht aber, dass nicht nur die CPU, sondern die komplette Hardware eines anderen Systems virtuell und möglichst exakt nachgebildet wird.

Beispiele:

- Es ist möglich, Software für andere Systeme zu entwickeln und zu testen. Es kann z. B. ein Programm für Palm OS auf einem PC entwickelt werden und mittels Palm Emulator getestet werden.

- Auf einem Linux-Rechner kann mit Hilfe von VMware ein PC emuliert werden, auf dem Windows installiert werden kann. Dadurch können Windows Anwendungen ausgeführt werden. Die meisten gekauften Windowsanwendungen können weiter eingesetzt werden.

- Ein Emulator erlaubt es, sich in Systeme einzuarbeiten, deren Anschaffung sonst sehr aufwändig wäre. Mit dem Hercules-Emulator wird z. B. auf einem PC ein S/370 emuliert, auf dem ein komplettes MVS (Multiple Virtual Storage) installiert wird.

- Weiters können alte Konsolenspiele aus den frühen achtziger Jahren dank geeigneter Emulatoren auf moderner Hardware laufen [3].

- Mit Emulatoren wie NistNet oder Dummynet können verschiedenste Netzwerkbedingungen im realen Netzwerk emuliert werden

2.2 NistNet

2.2.1 Allgemeines

Die offizielle Seite von NistNet ist unter dem Link [4] zu finden. Dort sind Installationshinweise, FAQs und eine genaue Beschreibung von NistNet ersichtlich. Auch können dort verschiedene Versionen von NistNet downgeloadet werden.

Nistnet wurde vom „National Institute of Standards and Technology“ (NIST) entwickelt. Der Emulator NistNet wird unter Linux auf einem Rechner der als Router arbeitet installiert. Dadurch können verschiedenste Netzwerkbedingungen im realen Netzwerk emuliert werden. Dabei greift NistNet direkt in den IP-forward-Mechanismus des Linux Kernels ein, wodurch das Verhalten der Verbindungen zwischen den angeschlossenen Rechnern von NistNet beeinflusst werden kann.

NistNet besteht im Wesentlichen aus zwei Teilen:

- Kernel Modul:

  Dieses wird in den Linux-Kernel geladen und beeinflusst den „networking“ und „real-time clock“ code von Linux.

- Einigen User-Interfaces:

  Diese werden verwendet, um die APIs zu konfigurieren und die Operationen des Kernelemulators zu kontrollieren.

Da NistNet auf einem Router installiert werden muss, werden auch mindestens zwei Netzwerkkarten benötigt, damit der Emulator zwischen zwei Subnetzwerke geschaltet ist. In dieser Arbeit wird von einem PC (Server) zu einen anderen PC (Client) - die sich beide jeweils in einem anderen Subnetzwerk befinden - der Netzwerkverkehr mittels NistNet beeinflusst, wobei der Router zwischen den beiden PCs positioniert ist. Eine genauere Beschreibung des Teststreckenaufbaus folgt in Kapitel 3.2. Auf dem Router wird mit Hilfe von NistNet der Netzwerkverkehr beeinflusst, wobei die IP-Adresse des Servers und des Clients und die entsprechende „delay-Zeit“ oder die entsprechende „drop-Rate“ angegeben wird. Die genaue Eingabe wird ebenfalls später noch erläutert.

NistNet stellt folgende Funktionalität zur Beeinflussung des Netzwerks zur Verfügung:

- Delay, Delsigma: Mittelwert und Standardabweichung der Verzögerungszeit in ms.
- Bandwidth: Maximal erlaubte Bandbreite in Bytes/Sekunde
- Drop: Prozentsatz der Verlustrate
- Dup: Prozentsatz der Paketduplizierungsrate

Dabei kann der Netzwerkverkehr zwischen der „Source-Adresse“ (IP-Adresse) und der „Destination-Adresse“ (IP-Adresse) beeinflusst werden.

2.2.2 Installation von NistNet

In dieser Arbeit wurde die NistNet Version 2.0.12 verwendet. Diese Version wurde im Juni 2002 veröffentlicht.

- Die Datei konnte als „nistnet.2.0.12.tar.gz“ File downgeloaded werden.
- Anschließend wurde mit dem Befehl

  `tar –xvzf nistnet.2.0.12.tar.gz` die gezippte Datei entpackt.
- Wechseln ins NistNet Verzeichnis:

  ```
  cd nistnet
  ```
- Starten der NistNet Konfiguration und erstellen des Makefiles

  ```
  ./configure
  ```

  Nach Eingabe von `./configure` muss nur den Aufforderungen der Konsole gefolgt werden.

  1.) Add explicit congestion notification (ECN) support: yes
  2.) Add class / type of service (COS) support: no
  3.) Which delay distribution do you want to use: experimental
- Installieren von NistNet, der API library und des user interfaces

  ```
  make
  make install
  ```

2.2.3 Anwendung von NistNet

Um mit NistNet arbeiten zu können, ist es erforderlich, als root angemeldet zu sein.

Damit NistNet gestartet werden kann, muss zuerst das Nistnet Emulator Modul in den Kernel geladen werden. Dies geschieht mit dem Befehl:

```
./Load.Nistnet
```

Grundsätzlich gibt es 3 verschiedene Arten um mit NistNet zu arbeiten.

- Cnistnet: neuer Kommandozeilen-Interpreter
- Hitbox: alter Kommandozeilen-Interpreter
- Xnistnet: GUI Version des User Interfaces

In dieser Arbeit wurde der neue Kommandozeilen-Interpreter „Cnistnet“ verwendet.

2.2.4 Cnistnet

Die anschließenden Befehle zu den Argumenten wurden direkt von der NistNet Seite übernommen [4].

```
-u
```
up (on)

```
-d
```
down (off)

```
-a src[:port[.protocol]] dest[:port[.prot]] [cos]
     add new
     [--delay delay [delsigma[/delcorr]]]
     [--drop drop_percentage[/drop_correlation]]
     [--dup dup_percentage[/dup_correlation]]
     [--bandwidth bandwidth]
     [--drd drdmin drdmax [drdcongest]]

-r src[:port[.prot]] dest[:port[.prot]] [cos]
```
remove

```
-s src[:port[.prot]] dest[:port[.prot]] [cos]
```
see stats

```
-S src[:port[.prot]] dest[:port[.prot]] [cos]
```
see stats continuously

```
[-n] -R
```
read current settings (-n numerical format)

```
-D value
```
debug on (value=0 none, 1 minimal,... 9 maximal)

```
-U
```
debug off

```
-G
```
global stats

```
-K
```
kickstart the clock

```
-F
```
flush the queues

```
-h
```
this help message

Auf die genaue Eingabe der Argumente wird später in Kapitel 3.4.4 eingegangen.

3 Teststrecke

3.1 Allgemeines

Die Teststrecke wurde vom Institut für Informatik zur Verfügung gestellt. Mit Hilfe dieser Teststrecke konnten die Datenströme zwischen den einzelnen Rechnern genau gemessen und anschließend analysiert werden. Grundsätzlich wird auf beiden Rechnern (PC 1 und PC 5) jeweils ein Videokonferenz Tool und ein Computerspiel installiert, wobei PC 5 als Server verwendet wird. Anschließend wird der Netzwerkverkehr zwischen den beiden PCs bestimmt. Die einzelnen Testszenarien werden später in diesem Kapitel noch detailliert beschrieben.

Der Teststreckenaufbau ist in Abbildung 3.1 zu sehen [5].

3.2 Aufbau der Teststrecke

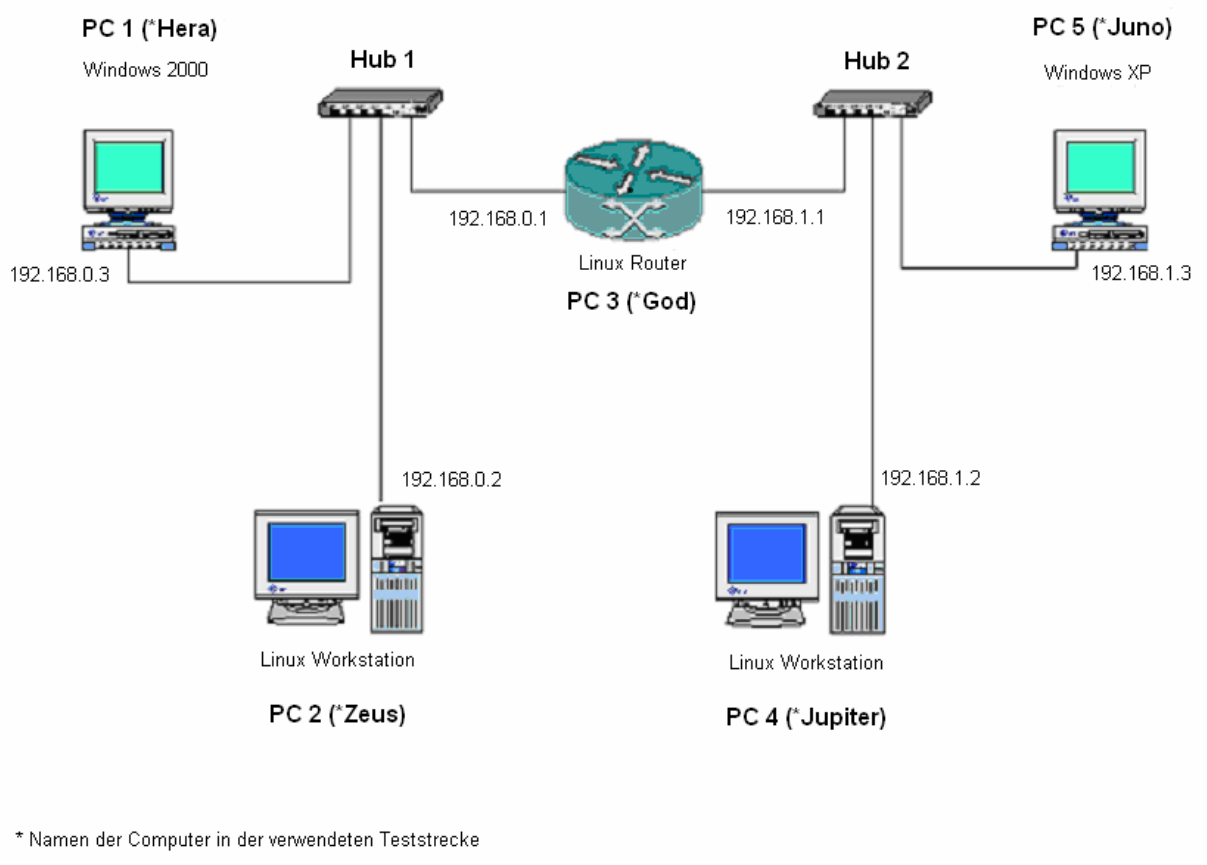

Abbildung 3.1 Teststreckenaufbau

3.3 Spezifikation der Teststrecke

3.3.1 Hardwarespezifikation

Die Hardwarespezifikation für die Rechner PC 1, PC 3 und PC 4 ist dieselbe. Die entsprechenden Daten sind anschließend angegeben.

Prozessor:	AMD Athlon XP 2200+
Mainboard:	VIA Apollo VT8366/A
Graphikkarte:	GeForce Ti 4400
Soundkarte:	VIA-AC'97
Netzwerkkarte:	Surecom EP320-R-100/10/M-PCI (100 MBit)
RAM:	512 MB DDR RAM

Die Hardwarespezifikation für Rechner PC 2 lautet:

Prozessor:	Pentium III – MMX
Graphikkarte:	3Dlabs Permedia2
Netzwerkkarte:	SMC EtherPower II 10/100
RAM:	130 MB

Die Hardwarespezifikation für Rechner PC 5 lautet:

Prozessor:	Intel Pentium III
Mainboard:	Intel Corporation 82443BX/7X
Graphikkarte:	NVIDIA RIVA TNT2/TNT2 Pro
Soundkarte:	Creative Sound Blaster PCI
Netzwerkkarte:	3Com EtherLink XL 10/100 PCI TX NIC
RAM:	256 MB SD RAM

3.3.2 Softwarespezifikation

PC 1:

Betriebssystem: Windows 2000

Verwendungszweck: Rechner auf dem die zu testende Applikation läuft

PC 2:

Betriebssystem: Red Hat Linux 7.1 Kernel 2.4.2-2

Verwendungszweck: Aufzeichnen des Netzwerkverkehrs zwischen PC 1 und PC 3.

PC 3:

Betriebssystem: Red Hat Linux 8.0 Kernel 2.4.18-24

Verwendungszweck: Router

PC 4:

Betriebssystem: Red Hat Linux 8.0 Kernel 2.4.18-24

Verwendungszweck: Markierungspakete von PC 4 zu PC 1 schicken und Aufzeichnen des Netzwerkverkehrs zwischen PC 5 und PC 1

PC 5:

Betriebssystem: Windows XP SP1

Verwendungszweck: Rechner auf dem die zu testende Applikation läuft, sowie „ping“ an PC1 schicken.

3.3.3 Verwendete Tools für die Durchführung der Test

3.3.3.1 Netzwerkspezifische Tools

Name:	tcpdump
Verwendungszweck:	Protokollierung des Netzwerkverkehrs
Firma:	Open Source
Version:	3.8
Quelle:	http://www.tcpdump.org/
Beschreibung:	tcpdump ist ein Programm um den gesamten Netzwerkverkehr eines Interfaces mitzuschneiden und zu analysieren (sog. Sniffer). Dies kann in vielen Fällen sehr nützlich sein, um z.B. Fehler in Netzwerkverbindungen zu finden. Die mitprotokollierten Daten lassen sich anschließend mit Hilfe von tcpdump sehr gut analysieren [6].

Name:	ping
Verwendungszweck:	um auftretende Verzögerungen zu beobachten
Firma:	Betriebssystemtool
Version:	-
Quelle:	auf jedem Rechner
Beschreibung:	Ping (Packet InterNet Groper) bezeichnet ein Programm, das mittels ICMP-Protokoll die Übertragungszeiten zwischen zwei verschiedenen Rechnern im Internet ermittelt. Dabei wird ein bestimmtes Paket (ICMP ECHO_REQUEST) vom absendenden Rechner an den Zielrechner versandt. Der Zielrechner schickt dieses Paket wieder zurück (daher "Echo"_Request). Aus Ankunftszeit - Absendezeit errechnet PING dann die Laufzeit.

Name:	mgen
Verwendungszweck:	Erzeugen eines definierten Hintergrundverkehrs
Firma:	Naval Research Laboratory
Version:	4.1
Quelle:	http://mgen.pf.itd.nrl.navy.mil/mgen.html
Beschreibung:	Multi-Generator (MGEN) ist ein Open Source Tool und wurde von der Naval Research Laboratory Gruppe entwickelt. Mit dem Tool mgen kann ein definierter UDP Netzwerkverkehr erzeugt werden. Dabei ist es möglich, die Anzahl der UDP-Pakete und den Zeitraum, in welchem diese Pakete an einen bestimmten Zielrechner gesendet werden sollen, anzugeben. Durch dieses Tool konnten bei jedem Testablauf die Zeitmarkierungspunke generiert werden [7].

Name:	NistNet
Verwendungszweck:	Netzwerkemulator
Firma:	National Institute of Standards and Technology
Version:	2.0.12

3.3.3.2 weitere Tools

Die anschließend erwähnten Tools dienen im Wesentlichen der graphischen Darstellung der Testergebnisse und der Dokumentation der Arbeit und werden daher nicht genauer erläutert.

Gnuplot:

Wurde zur graphischen Darstellung der Testergebnisse verwendet.

Active Perl:

Mit diesem Programm wurden die Perlskripte ausgeführt.

Real One Player:

Abspielen des Matrix Testvideos auf PC 5.

Microsoft Word:

Wurde zur schriftlichen Dokumentation dieser Arbeit verwendet.

Adobe Acrobat:

Konvertierung der Arbeit in das pdf Format.

3.4 Skripte

Bei dieser Arbeit wurden mgen-, Perl- und Gnuplot-Skripte [5] verwendet. Weiters wurden NistNet Skripte verwendet. Der Verwendungszweck der einzelnen Skripte soll anschließend kurz erklärt werden.

- mgen – Skripte: um Markierungspunkte für die jeweiligen Zeitintervalle zu erzeugen
- Perl – Skripte: um die erhaltenen Messergebnisse auszuwerten
- Gnuplot – Skripte: um die aufbereiteten Messergebnisse graphisch darzustellen
- NistNet – Skripte: um die verschiedenen „delay-Zeiten“ und „drop-Raten“ zu emulieren

3.4.1 mgen – Skripte

Mgen wurde in den Versuchen für das Setzen von timestamps verwendet. Mithilfe von mgen wurde der Zeitbereich der einzelnen Versuche in 3 Abschnitte eingeteilt. Im ersten Bereich wurde weder ein delay noch ein drop erzeugt. Im zweiten Bereich wurde mit Hilfe von NistNet ein delay oder drop erzeugt. Am Beginn des dritten Bereiches wurde das delay oder drop wieder entfernt.

Hier zur Verdeutlichung das „notraffic.mgen“ Script:

```
################################################
# MGEN script
################################################


0.0 ON 10 UDP SRC 5001 DST 192.168.0.3/5000 PERIODIC [1 100]
30.0 ON 11 UDP SRC 5001 DST 192.168.0.3/5000 PERIODIC [1 100]
90.0 ON 12 UDP SRC 5001 DST 192.168.0.3/5000 PERIODIC [1 100]
120.0 ON 13 UDP SRC 5001 DST 192.168.0.3/5000 PERIODIC [1 100]

0.1 OFF 10
30.1 OFF 11
90.1 OFF 12
120.1 OFF 13
```

Es werden je vier Transmission Events (Übertragungsereignisse) mit den FlowId´s 10, 11, 12, 13 definiert. Dabei sind die vier Events jene, die 100 Byte Markierungspakete schicken. Das heißt in der 0.ten, 30.ten, 90.ten und 120.ten Sekunde wird jeweils ein Markierungspaket (timestamp) geschickt.

Folgendes Script wurde für mgen erzeugt:

- **notraffic.mgen**
  Kein Hintergrundverkehr, es werden nur die Markierungspakete (timestamps) gesendet.

3.4.2 Perl – Skripte

Um die mit tcpdump erzeugten Outputfiles zu parsen, wurden Perlscripts eingesetzt. Das Ziel war dabei die Outputfiles so zu parsen, dass diese von Gnuplot visualisiert werden konnten.

Folgende Perlscripts wurden verwendet:

- divideproto.pl

  Dem Script wird als Argument das zu parsende tcpdump outputfile übergeben. Dieses wird dann nach verschiedenen Kriterien geparst und es werden verschieden Files generiert.

  UDP100
  Hier werden alle UDP Pakete gespeichert, die der Server zum Client schickt.

  UDP250
  Hier werden alle mgen Pakete gespeichert.

  UDPRest
  Hier werden alle anderen UDP Pakete gespeichert

  RTT
  Hier werden die ping Pakete gespeichert, berechnet als RTT.

  Die weiteren drei Files TCP100, TCP250 und TCPRest waren nicht von Bedeutung, da kein TCP Verkehr beobachtet wurde.

- **parseTraffic.pl**

  Die von divideproto.pl erzeugten Files, außer RTT, können dann gnuplotgerecht weiter geparst werden. Dem Script werden je zwei Argumente übergeben: das zu parsende File und der Name des zu erstellenden Files. So wird ein File im folgenden Format erstellt:

1.Spalte	Zeit in Sekunden
2.Spalte	Durchsatz in KByte/s
3.Spalte	Anzahl Pakete pro Sekunde

3.4.3 Gnuplot – Skripte

- throughput.plt

  Visualisiert den Durchsatz der UDP – Pakete, die vom Sender gesendet werden und derer, die beim Empfänger ankommen. Dabei werden 2 Kurven (server send und client receive) übereinander gelegt. So kann visualisiert werden, wie die Anwendungen reagieren und wie viele Pakete verloren gehen.

- ping.plt

  Visualisiert die Round Trip Time die mit ping gemessen wurde.

- packetInSec.plt

  Visualisiert die in der Sekunde gesendeten UDP – Pakete vom Sender.

- avPacketLenght.plt

  Visualisiert die durchschnittliche UDP – Paketgröße, die in der Sekunde vom Sender übermittelt wird.

3.4.4 NistNet – Skripte

Um die „delay-Zeiten“ und „drop-Raten“ durch NistNet hinzuzufügen und zu entfernen, wurden zwei Skripte erzeugt. Dabei können in NistNet Tabelleneinträge hinzugefügt und entfernt werden. Es muss jeweils eine Source-Adresse und eine Destination-Adresse sowie die „delay-Zeit“ (in ms) oder die „drop-Rate“ (in %) angegeben werden.

- nistdelay.sh

  Dadurch werden „delay-Zeiten“ in NistNet auf dem Router hinzugefügt und entfernt.

- nistdrop.sh

  Dadurch werden die „drop-Raten“ in NistNet auf dem Router hinzugefügt und entfernt.

Zur Verdeutlichung wird hier ein „nistdelay.sh“ Skript angegeben:

```
sleep 30
cnistnet -a 192.168.1.3 192.168.0.3 --delay 50
sleep 60
cnistnet -r 192.168.1.3 192.168.0.3 --delay 50
```

Hier wird nach 30 Sekunden ein Delay von 50ms zwischen der Source-Adresse 192.168.1.3 und der Destination-Adresse 192.168.0.3 eingerichtet. Nach anschließenden 60 Sekunden wird dieses Delay wieder entfernt. Die gesamten verwendeten Skripte können der beiliegenden CD entnommen werden.

3.5 Beschreibung des Testablaufes

3.5.1 Videokonferenz

Um eine Videokonfernz zu simulieren, musste jeweils auf den Rechnern PC 1 und PC 5 das gleiche Videokonferenz Tool gestartet werden. Um immer den gleichen Netzwerkverkehr (Reproduzierbarkeit) bei jedem Tool zur Verfügung zu haben, wurde auf PC 5 mittels Real One Player das File „test_stream_max512.rm“, welches ein aus einem Zusammenschnitt mehrerer schneller Szenen des Matrix Reloaded Trailers bestehendes Videofile darstellt, abgespielt. Dieses Videofile wurde aus der Arbeit „Analyse des Netzwerkverhaltens von Videostreaming im Internet“ [5] entnommen. Um die graphischen Daten auf PC 1 zu übertragen, wurde der gesamte Bildschirm von PC 5 mittels einer Web-Cam gefilmt. Dadurch konnten die bewegten Bilder einer Videokonferenz simuliert werden. Der Ton einer Videokonferenz konnte ebenfalls mit Hilfe des Files „test_stream_max512.rm“ auf PC 1 übertragen werden. Dies erfolgte in der Weise, dass die Daten vom Audioausgang des PC 5 auf dessen Mikrophoneingang umgelenkt und somit auf PC 1 übertragen wurden. Das heißt, die graphischen Daten werden mittels einer Web-Cam und die Audiodaten werden direkt vom Mikrophoneingang des PC 5 auf PC 1 übertragen. Für die einzelnen Testfälle waren immer nur die ersten 120 Sekunden des Videofiles von Bedeutung.

3.5.2 iVisit

iVisit funktioniert im Grunde wie ein erweiterter Chat-Client. Das Programm überträgt Bild und Ton in Echtzeit, unterstützt aber auch ganz klassische Textnachrichten. In der vorangegangen Arbeit wurde die iVisit Version 3.1.5 verwendet. Da aber die Versionen des Videokonferencing Tools ständig aktualisiert werden und am Beginn der Videokonferenz eine Verbindung mit dem iVisit Server aufgebaut wird, konnte diese Version nicht mehr verwendet werden, da der Server nicht beliebig alte Versionen von iVisit zulässt. Daher wurde in dieser Arbeit die Version 3.3.4 verwendet.

3.5.3 Computerspiele

Hier wurde ebenfalls jeweils das gleiche Spiel auf den Rechnern PC 1 und PC 5 gestartet. Dabei wurde zuerst der Rechner PC 5 als Spiel-Server gestartet und anschließend durch direkte Eingabe, auf PC 1 (Client), der IP-Adresse zum Server (PC 5) verbunden.
Bei den zwei getesteten Spielen wurde jeweils eine Spielsituation simuliert, wie sie bei üblichen Internetspielen auftritt. Das heißt es herrschte jeweils normaler Spielfluss.

3.5.4 Fifa Football 2004

Fifa Football 2004 ist eine Simulation für den echten Fußball-Fan. Das Release erschien am 30. Oktober 2003 von Electronic Arts. Dieses Spiel wurde von den Kanadiern der Firma EA-SPORTS für mehrere Plattformen entwickelt: Playstation2, Xbox, Gamecube und PC [8].

Die jeweiligen Testszenarien der verschiedenen „delay-Zeiten“ und „drop-Raten“ wurden jeweils mit einem Spielverkehr mit normalem Spielfluss durchgeführt.

In Abbildung 3.2 ist ein Ausschnitt eines solchen Testszenarios ersichtlich.

Abbildung 3.2: Fifa Football 2004

3.5.5 Jedi Knight II – Jedi Outcast:

Jedi Knight II – Jedi Outcast ist ein First-Person-Shooter Spiel, das auf den Filmen Star Wars des Regisseurs George Lucas basiert. Das Spiel wurde im Frühjahr 2002 von Lucasarts veröffentlicht [9].

Die jeweiligen Testszenarien der verschiedenen „delay-Zeiten“ und „drop-Raten“ wurden jeweils mit einer Gesamtspielerzahl von 18, davon 16 Bots und 2 reale Spieler durchgeführt.

In Abbildung 3.3 ist ein Ausschnitt eines solchen Testszenarios ersichtlich.

Abbildung 3.3: Jedi Knight II

Abbildung 3.4: „Testablauf und Datenflussrichtung“ zeigt eine prinzipielle Struktur des ersten Testverfahrens und des Verkehrsflusses. Anschließend wird sich aber zeigen, dass mit dieser Messanordnung falsche Messdaten ermittelt worden sind und daher die Messanordnung geringfügig verändert werden musste.

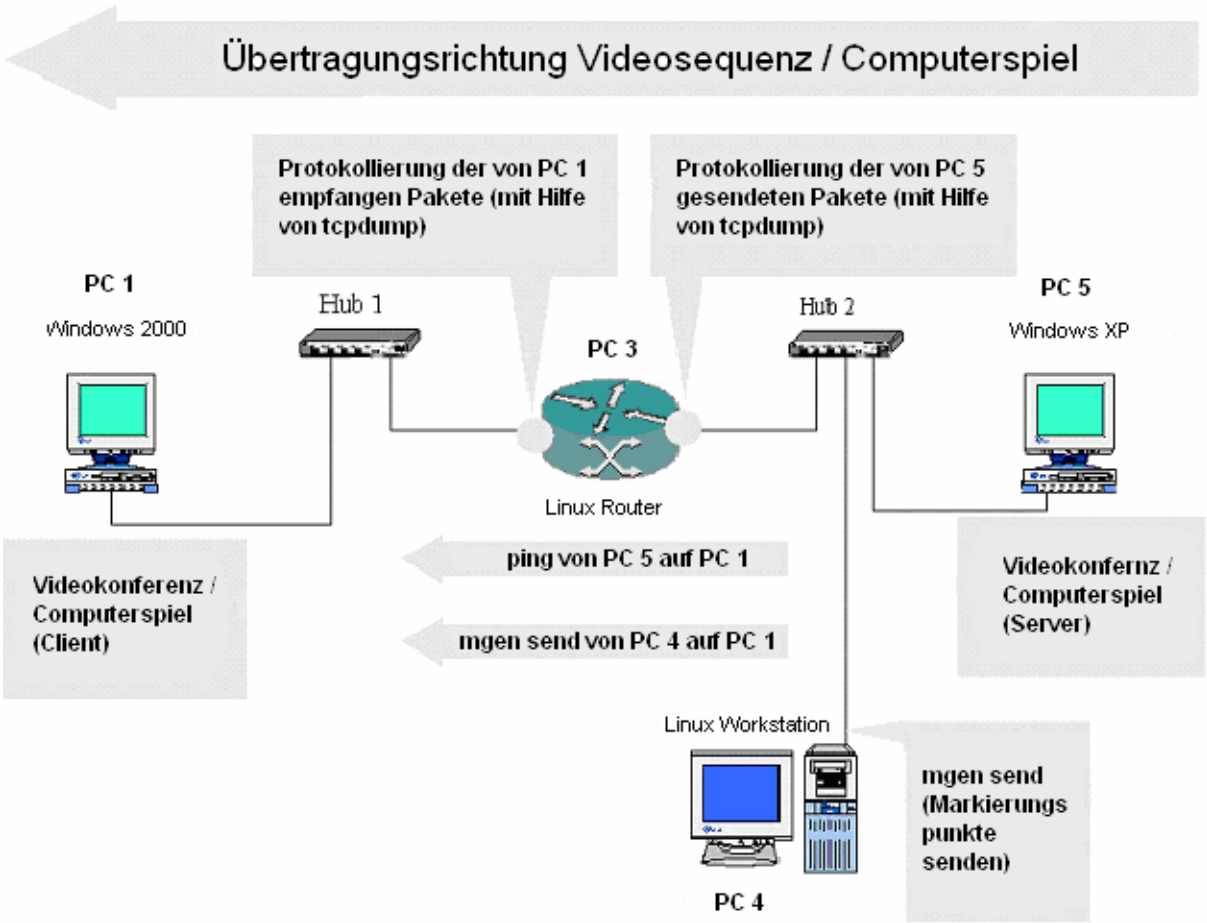

Abbildung 3.4: Testablauf und Datenflussrichtung

3.5.6 Ablaufbeschreibund des ersten Testablaufes

1. Auf beiden Windows Rechnern (Windows XP und Windows 2000) muss die zu testende Videokonferenz oder das Computerspiel gestartet und eine Verbindung zwischen PC 1 und PC 5 in der jeweiligen Anwendung aufgebaut werden.

2. tcpdump auf PC 3 mit dem Aufruf `tcpdump –i eth1 –vvttnn > tcpout.txt` in der shell starten.

3. tcpdump auf PC 3 mit dem Aufruf `tcpdump –i eth2 –vvttnn > tcpout.txt` in der shell starten.

4. Der 4. Schritt wird nur bei der Untersuchung des Videokonferenz-Tools ausgeführt.

   Das File „test_stream_max512.rm“ auf PC 5 abspielen und mittels einer Web-Cam den gesamten Bildschirm filmen. Zusätzlich mit einem Koaxialkabel die analogen Audiodaten vom Audioausgang auf den Mikrophoneingang von PC 5 umlenken.

5. ping von PC 5 auf PC 1 starten: mit dem Aufruf `ping –n 10000 192.168.0.1` in der shell starten. (ping –n 10000 192.168.0.1 führt auf dem Windows Rechner 10000 pings auf Rechner PC 1 aus).

6. Das jeweilige Skript für NistNet, je nachdem ob eine Verzögerung oder ein Verlust emuliert werden soll, auf PC 3 mit dem Aufruf `./nistdelay.sh` (für Zeitverzögerung) oder `./nistdrop.sh` (für Paketverlust) starten.

7. mgen auf PC 4 mit dem Script Aufruf `mgen input notraffic.mgen` in der shell starten.

8. Nach 120 Sekunden beendet mgen auf PC 4 die Aktion. Danach können alle Anwendungen gestoppt bzw. beendet werden.

Nach einigen Versuchen durch Emulation einer Verzögerungszeit mittels NistNet und anschließender Auswertung durch Gnuplot stellte sich heraus, dass die Messergebnisse nicht korrekt sein können, da der Throughput der von PC 5 erzeugt wurde, über den gesamten Messbereich betrachtet, größer war als der, der von PC 1 empfangen wurde. Bei Emulation einer Verzögerungszeit, sollte aber der gesendete und empfangene Throughput, über den gesamten Messbereich gesehen, den gleichen Wert aufweisen.

Zur Verdeutlichung des gemessenen Throughputs (gesendeter und empfangener Throughput) bei Emulation einer Verzögerungszeit von 5 ms (beim Computerspiel Fifa Football 2004) durch NistNet wird dieser in Abbildung 3.5 dargestellt.

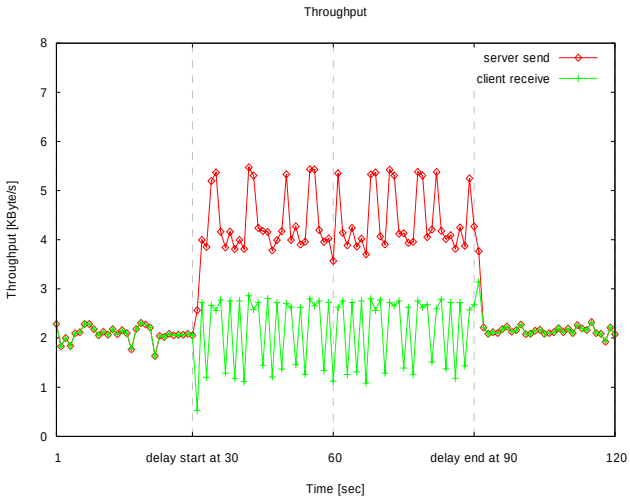

Abbildung 3.5: Throughput

In Abbildung 3.5 ist deutlich zu erkennen, dass der Throughput, der gesendet wurde, um einiges höher ist als der Throughput, der empfangen wurde.
Diese Fehlmessung konnte darauf zurückgeführt werden, dass die gesendeten und empfangenen Pakete mittels tcpdump jeweils an den Netzwerkkarten (eth1 und eth2) des Routers (PC 3) gemessen wurden. Da NistNet direkt in den Kernel des Routers eingreift,

wurden die Messergebnisse verfälscht. Auf Grund dieser Tatsache sind anschließend die gesendeten und empfangenen Pakete mit einem zusätzlichen Linux Rechner ermittelt worden. Es konnte festgestellt werden, dass nach der neuen Messmethode der Throughput, der gesendet wurde, auch dem Throughput, der empfangen (über dem gesamten Messbereich gesehen) wurde, entsprach. Durch die neue Messanordnung wurde ein zusätzlicher Rechner (PC 2) zur Teststrecke hinzugefügt, auf dem Linux installiert werden musste.

Abbildung 3.6: „Veränderter Testablauf und Datenflussrichtung“ zeigt eine prinzipielle Struktur des neuen Messaufbaus und des Verkehrsflusses.

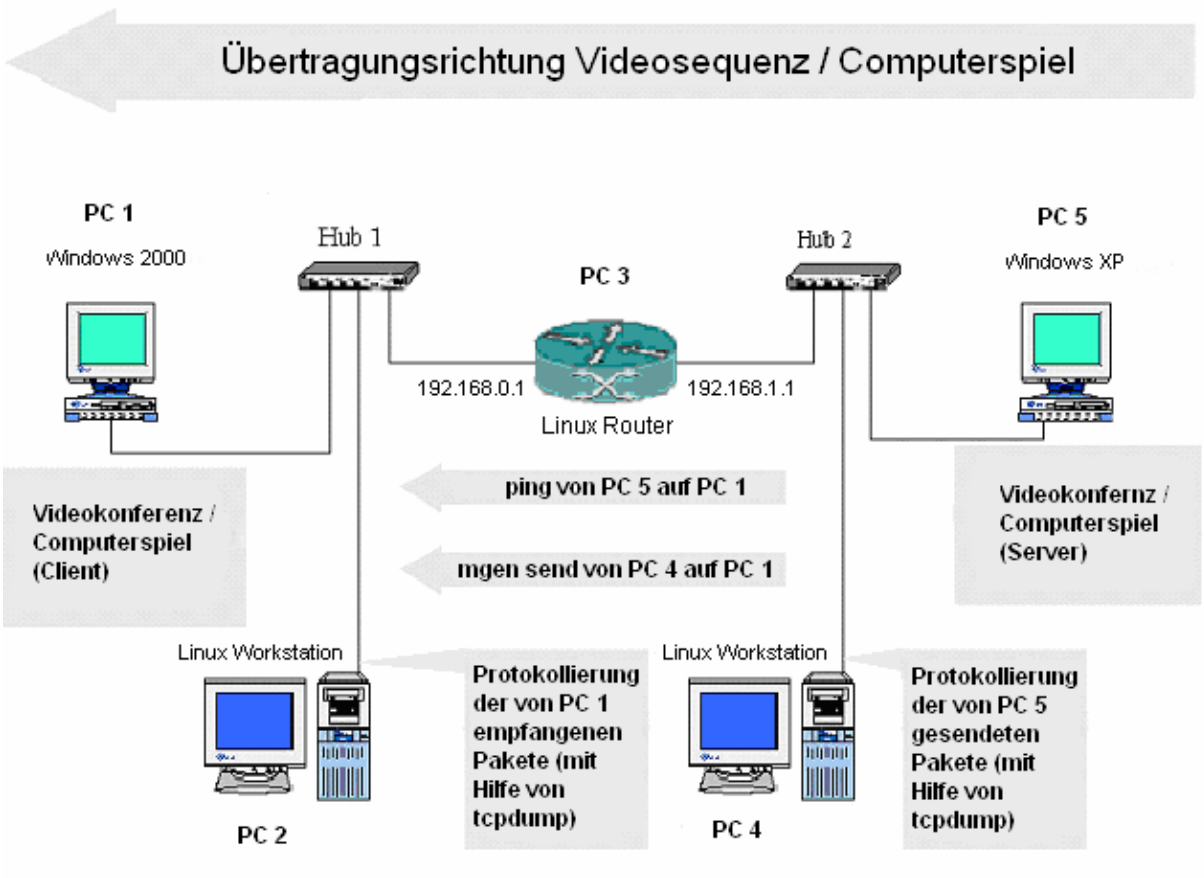

Abbildung 3.6: Veränderter Testablauf und Datenflussrichtung

3.5.7 Ablaufbeschreibung des veränderten Messaufbaus

1. Auf beiden Windows Rechnern (Windows XP und Windows 2000) muss die zu testende Videokonferenz oder das Computerspiel gestartet und eine Verbindung zwischen PC 1 und PC 5 in der jeweiligen Anwendung aufgebaut werden.

2. tcpdump auf PC 2 mit dem Aufruf `tcpdump –i eth0 –vvttnn > tcpout.txt` in der shell starten.

3. tcpdump auf PC 4 mit dem Aufruf `tcpdump –i eth0 –vvttnn > tcpin.txt` in der shell starten.

4. Dieser Schritt wird nur bei der Untersuchung des Videokonferenz-Tools ausgeführt.

   Das File „test_stream_max512.rm“ auf PC 5 abspielen und mittels einer Web-Cam den gesamten Bildschirm filmen. Zusätzlich mit einem Koaxialkabel die analogen Audiodaten vom Audioausgang auf den Mikrophoneingang von PC 5 umlenken.

5. ping von PC 5 auf PC 1 starten: mit dem Aufruf `ping –n 10000 192.168.0.1` in der shell starten. (ping –n 10000 192.168.0.1 führt auf dem Windows Rechner 10000 pings auf Rechner PC 1 aus)

6. Das jeweilige Skript für NistNet, je nachdem ob eine Verzögerung oder ein Verlust emuliert werden soll, auf PC 3 mit dem Aufruf `./nistdelay.sh` (für Zeitverzögerung) oder `./nistdrop.sh` (für Paketverlust) setzen

7. mgen auf PC 4 mit dem Script Aufruf `mgen input notraffic.mgen` in der shell starten

8. Nach 120 Sekunden beendet mgen auf PC 4 die Aktion. Danach können alle Anwendungen gestoppt bzw. beendet werden.

Die beiden tcpdump Outputfiles tcpin.txt und tcpout.txt stehen auf PC 4 und PC 2 zur Verfügung. Dabei sind in tcpin.txt die vom Sender der festgelegten Videosequenz / Computerspiels (PC 5) gesendeten Pakete und in tcpout.txt die von PC 1 empfangenen Pakete protokolliert. Diese beiden Files können entsprechend geparst und dann mit dem Programm Gnuplot dementsprechend visualisiert werden. Die entsprechenden Gnuplot Skriptfiles wurden bereits unter Punkt 3.4.3 näher erläutert.

4 Diskussion der Testergebnisse

Anschließend sollen die graphischen Testergebnisse genauer diskutiert werden. Es sind die Programme

- Fifa Football 2004
- Jedi Knight II
- iVisit (V.3.3.4)

getestet worden. Es wurden bei jeder Anwendung jeweils mit NistNet auf dem Router einige „delay-Zeiten“ und „drop-Raten“ emuliert. Die Computerspiele und das Videoconferencing Tool wurden dabei auf PC 5 und PC 1 ausgeführt.

Im folgendem sind die wesentlichsten Testszenarien und die Diagramme

- sum throughput
- sum number of packets
- average packet length

der einzelnen Anwendungen beschrieben. Weiters befinden sich im Anhang dieser Arbeit einige graphische Auswertungen der einzelnen Testszenarien, in denen der Durchsatz, die Round Trip Time, die Pakete pro Sekunde und die durchschnittliche Paketlänge pro Sekunde für jedes Testszenario einzeln dargestellt ist.
Die Daten der gesamten Testszenarien können der beiliegenden CD entnommen werden.

4.1 Fifa Football 2004

4.1.1 delay Fifa Football 2004

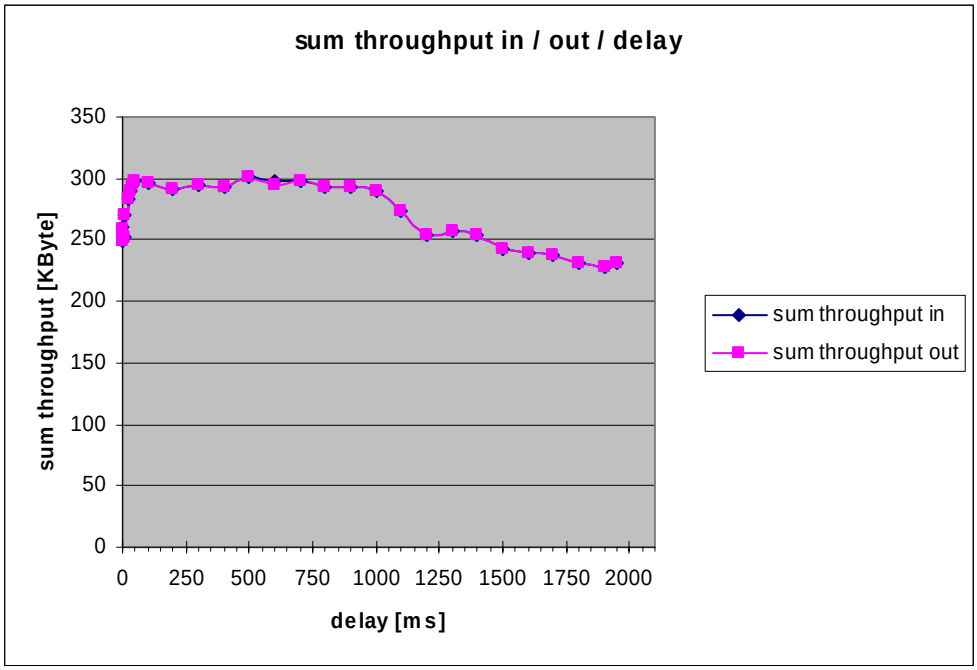

Abbildung 4.1: sum throughput / delay fifa football

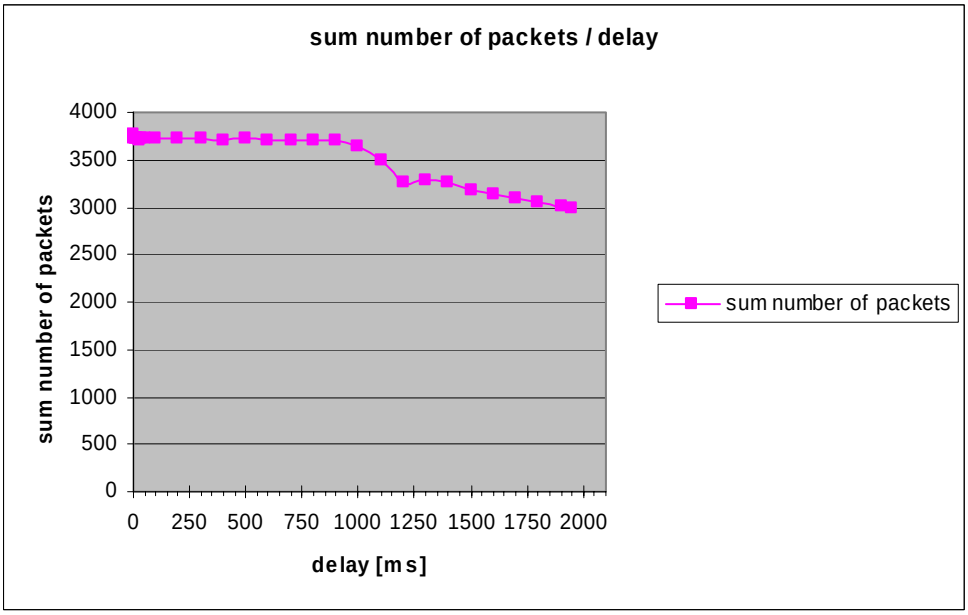

Abbildung 4.2: sum number of packets / delay fifa football

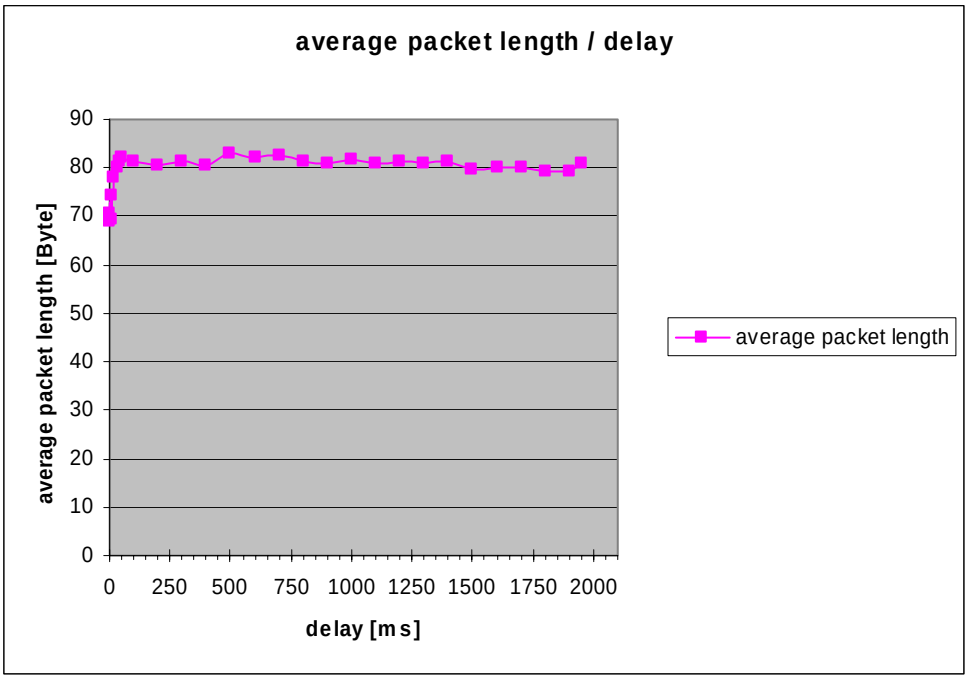

Abbildung 4.3: average packet length / delay fifa football

Bei Emulation einer Verzögerungszeit durch NistNet ist in Abbildung 4.1 (sum throughput / delay fifa) zu sehen, dass die Summe des Durchsatzes von einer Verzögerungszeit von 0ms bis zu einer Verzögerungszeit von ca. 40ms von ca. 250KByte bis zu ca. 300KByte ansteigt. Ab diesen 40ms bis zu ca. 1000ms bleibt die Summe des Durchsatzes annähernd konstant (ca. 300KByte). Wird hingegen die Verzögerungszeit weiter erhöht, sinkt auch die Summe des Durchsatzes stetig und erreicht bei einer Verzögerungszeit von 1950ms bei ca. 230KByte ihren kleinsten Wert. Bei einer Verzögerungszeit größer als 1950ms bricht die Verbindung des Clients ab und das Spiel kann nicht mehr fortgesetzt werden.

Auch in Abbildung 4.2 (sum number of packets / delay fifa) kann ein ähnliches Verhalten festgestellt werden. Die Summe der Pakete bleibt bis zu ca. 1000ms auf einen annähernd konstanten Wert (3700 Pakete). Bei einer Zeitverzögerung über 1000ms sinkt die Paketanzahl ständig ab und erreicht bei 1950 mit 3000 Packet ihren niedrigsten Wert. In Abbildung 4.3 (average packet length / delay fifa) hingegen zeigt sich ein anderes Bild. Hier ist ersichtlich, dass die durchschnittliche Paketlänge bis zu einer Zeitverzögerung bis zu ca. 40ms ansteigend ist. Bei einer weiteren Erhöhung der Zeitverzögerung liegt die durchschnittliche Paketlänge im Wesentlichen konstant bei 80 Byte.

Weiters zeigen die Diagramme, die sich im Anhang befinden, (Diagr.F.d.0ms.1 bis Diagr.F.d.1950ms.4), dass sich der Durchsatz mit zunehmender Zeitverzögerung (ab der 30. Sekunde bis zur 90. Sekunde) erhöht. Dies ist bei einer Verzögerungszeit von 1000ms (Diagr.F.d.1000ms.1) besonders gut zu erkennen. Auch kann man in diesem Diagramm bei einer Zeitverzögerung von 1000ms sehr gut erkennen, dass bei diesem Wert der Client im ersten Moment (bei Beginn der Zeitverzögerung in der 30. Sekunde) kaum noch Daten empfängt, wobei am Ende der Zeitverzögerungsperiode (in der 90. Sekunde) der Client genau um diesen Wert mehr empfängt. Weiters ist auffällig, dass bei einem größeren Zeitverzögerungswert der Durchsatz während der Zeitverzögerungsperiode enorm zu schwanken beginnt (Diagr.F.d.1950ms.1). Auch die Anzahl der gesendeten Pakete pro Sekunde beginnt ab einer größeren Zeitverzögerung zu schwanken und wird kleiner

(Diagr.F.d.1950ms.3). Die durchschnittliche Paketlänge pro Minute wird während der Zeitverzögerungsperiode erhöht, wobei ab einem gewissen Wert der Zeitverzögerung (am Ende dieser Periode) die durchschnittliche Paketlänge pro Minute sprunghaft ansteigt aber sofort wieder abnimmt (Diagr.F.d.1500ms.4).

4.1.2 drop Fifa Football 2004

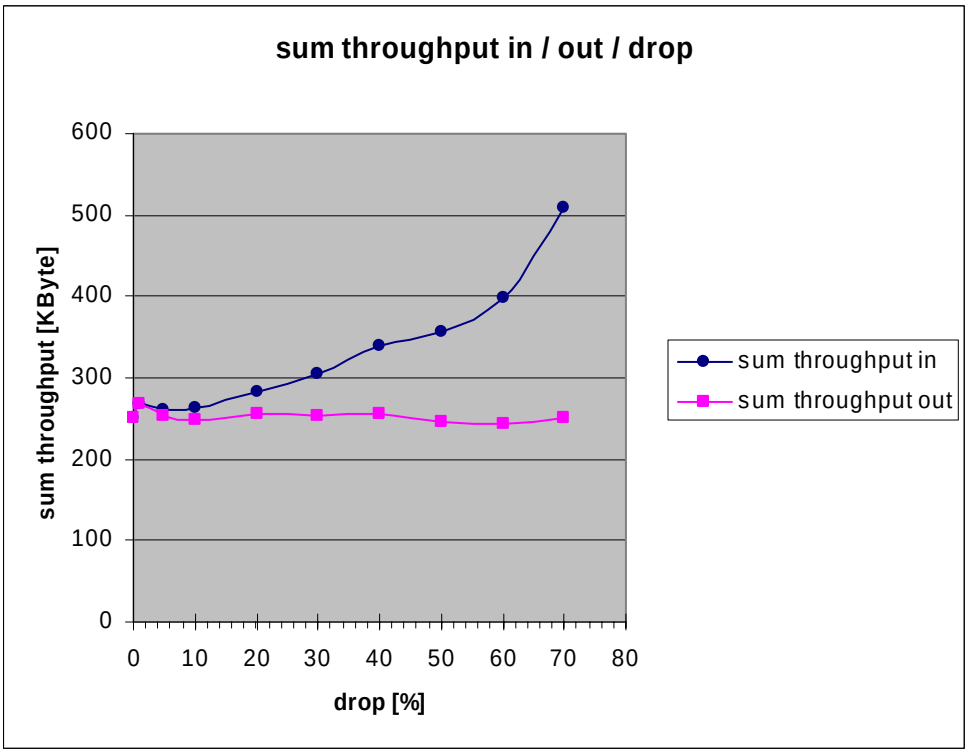

Abbildung 4.4: sum throughput in / out / drop fifa football

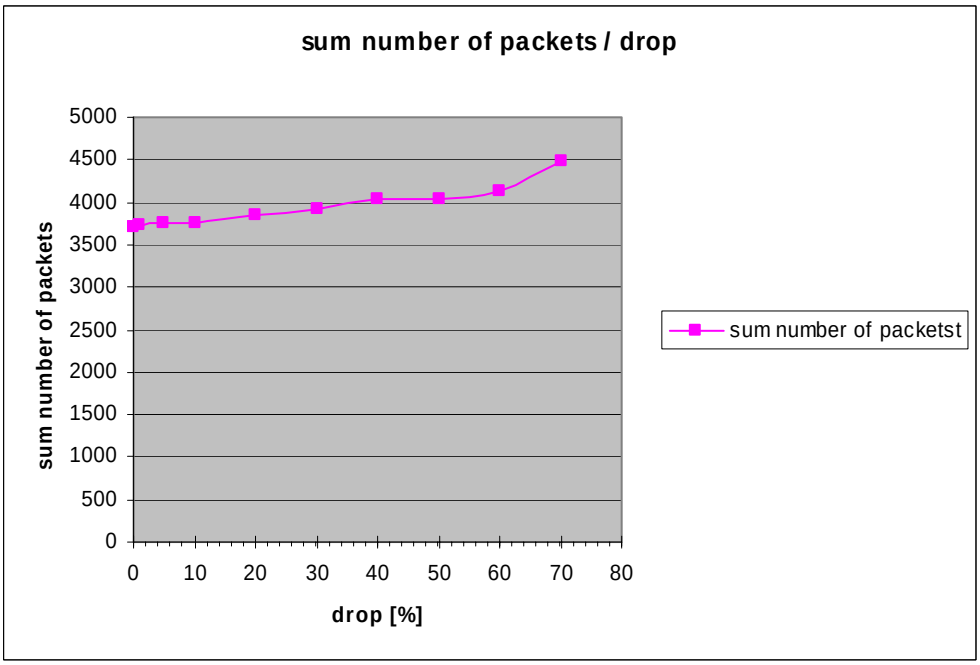

Abbildung 4.5: sum number of packets / drop fifa football

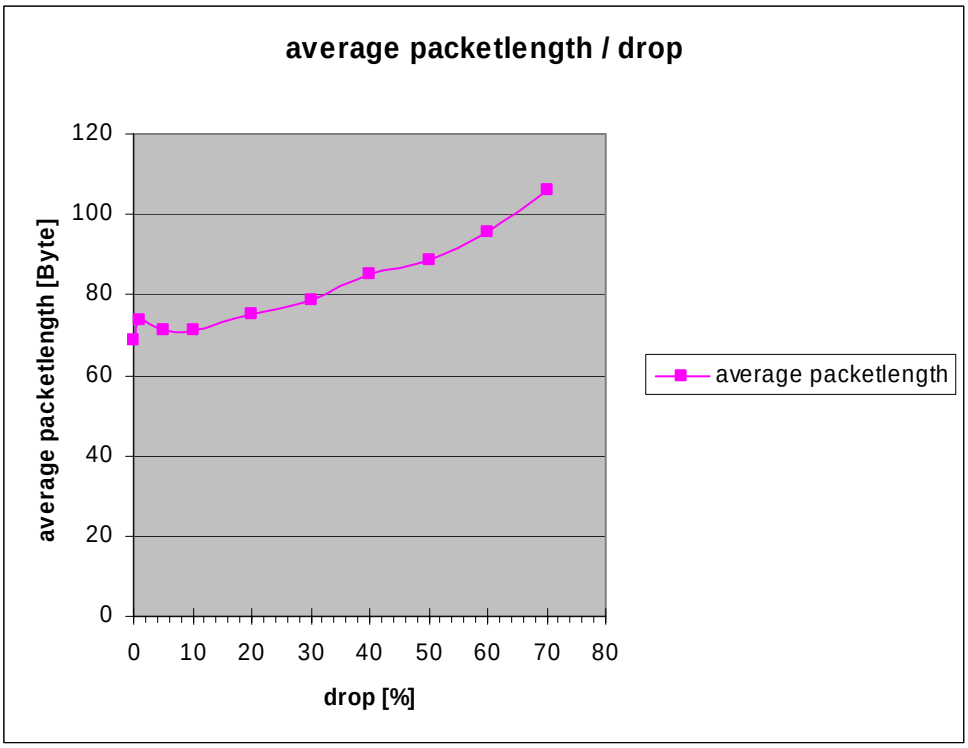

Abbildung 4.6: average packet length / drop fifa football

Bei Emulation von verschiedenen Verlustraten durch NistNet ist in Abbildung 4.4 die Summe der gesendeten Pakete des Servers und die Summe der empfangenen Pakete des Clients zu sehen. Hier ist ersichtlich, dass die Summe des Durchsatzes, den der Client empfängt, über den gesamten Testbereich im Wesentlichen konstant bleibt (ca. 250KByte). Um dies zu gewährleisten, muss aber der Server den Durchsatz je nach Verlustrate erhöhen. Die Summe des Durchsatzes der gesendeten Pakete erreicht sein Maximum mit ca. 500KByte. Bei einer Verlustrate von ca. 70% bricht die Verbindung zum Client ab. Die Summe der Anzahl der Pakete wird zwischen einer Verlustrate von 0 bis 70% nur leicht erhöht (Abbildung 4.5). Die durchschnittliche Paketlänge zeigt in Abbildung 4.6 beinahe einen linearen Anstieg bei einer Verlustrate zwischen 0 und 70% (ca. 70Byte bis 105Byte).

In den Diagrammen (F.d.0%.1 bis F.d.70%.3) ist ebenfalls ersichtlich, dass zwischen der 30. und der 90. Sekunde (Emulation verschiedener Verlustraten) der Durchsatz, der vom Server gesendet wird, in diesem Bereich je nach Verlustrate erhöht wird, während der Durchsatz der vom Client empfangen wird im Wesentlichen konstant bleibt. Die Pakete pro Sekunde und die durchschnittliche Paketlänge pro Sekunde werden in dem Zeitraum von 30 und 90 Sekunden ebenfalls erhöht (Diagr. F.d.70%.2 und Diagr. F.d.70%.3).

4.2 Jedi Knight II

4.2.1 delay Jedi Knight II

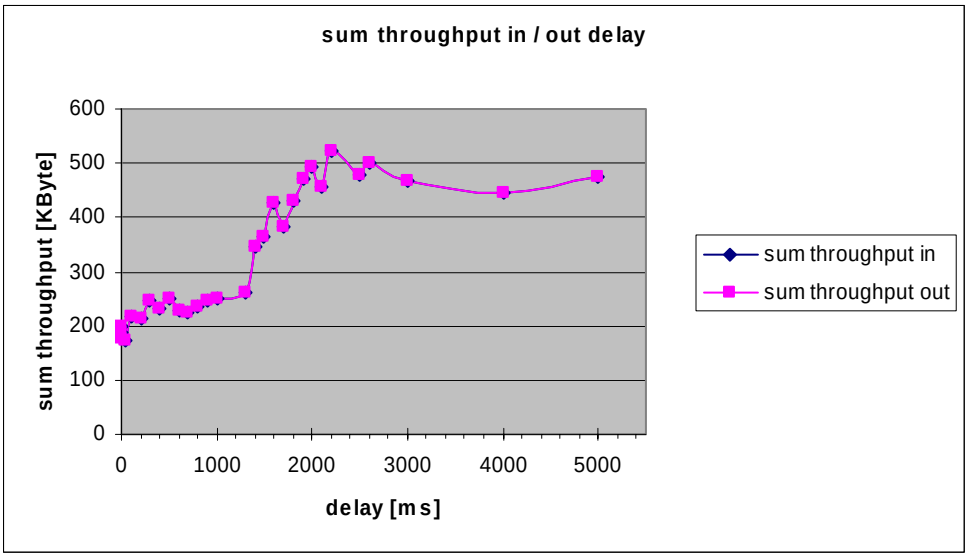

Abbildung 4.7: sum throughput in / out / delay jedi knight

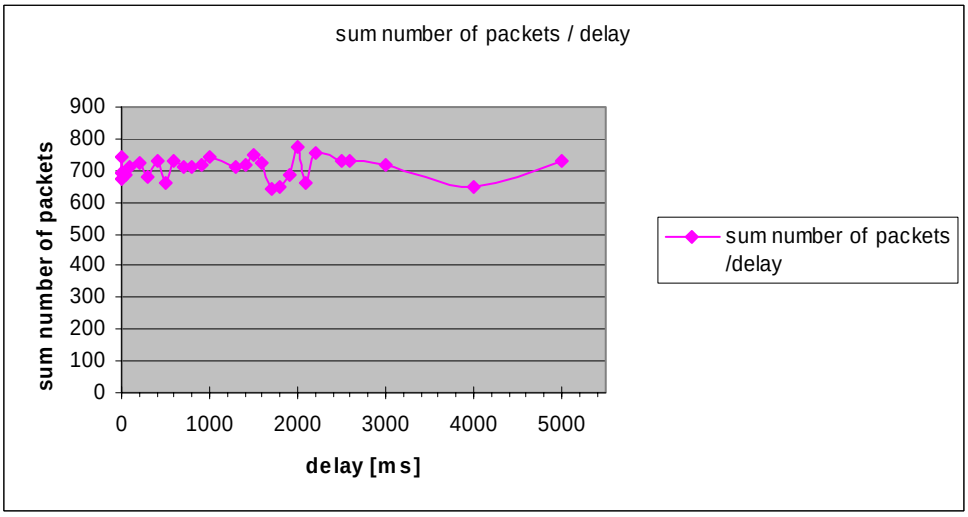

Abbildung 4.8: sum number of packets / delay jedi knight

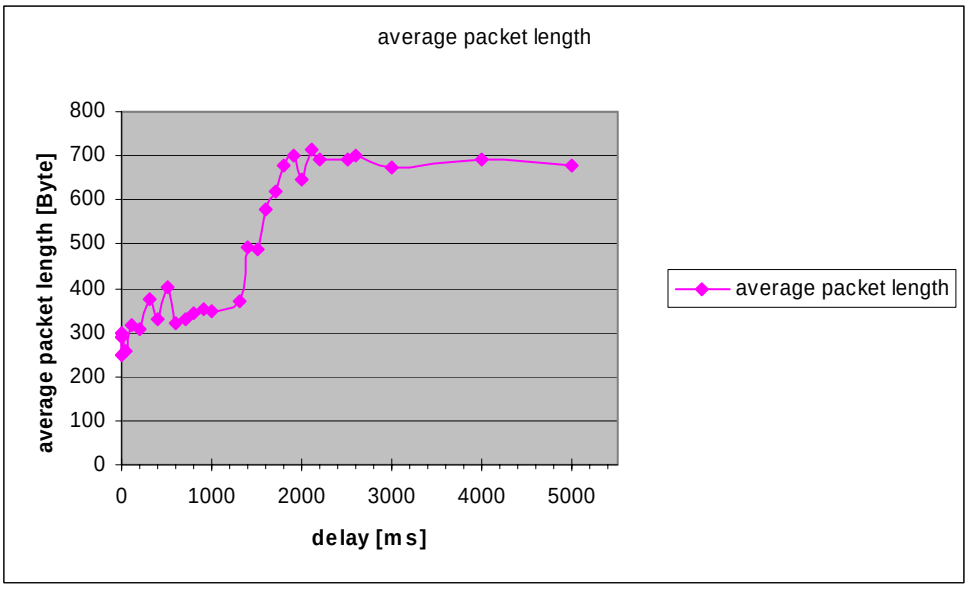

Abbildung 4.9: average packet length / delay jedi knight

Die Summe des Durchsatzes des Computerspiels Jedi Knight II liegt während einer Zeitverzögerung von ca. 0 bis 1300ms auf einem beinahe konstanten Wert von ca. 200KByte (Abbildung 4.7). Ab einer Zeitverzögerung von 1300ms steigt die Summe des Durchsatzes stetig an. Der Versuch wurde bis zu einer Zeitverzögerung von 5000ms durchgeführt, bei dem der Wert bei ca. 480KByte liegt. Das Diagramm (Abbildung 4.9) „durchschnittliche Paketlänge“ zeigt ein ähnliches Verhalten. Die Summe der Anzahl der Pakete (Abbildung 4.8) hingegen weist während des gesamten Versuchszeitraums einen beinahe konstanten Wert auf.

Auch die Diagramme (Diagr. J.d.0ms.1 bis Diagr. J.d. 5000ms.4) zeigen, dass der Durchsatz während des Zeitraums der Zeitverzögerung (zwischen 30 und 90 Sekunden) je nach deren Wert ansteigend ist (Diagr. J.d.2000ms.1). Auch ist ersichtlich, dass die Anzahl der Pakete während des gesamten Versuchzeitraums relativ konstant bleibt (Diagr. J.d.2000ms.3 und Digr. J.d.5000ms.3). Die durchschnittliche Paketlänge steigt ebenfalls je nach Zeitverzögerung (im Zeitraum zwischen 30 und 90 Sekunden) an (Diagr. J.d.2000ms.3 und Digr. J.d.5000ms.3).

4.2.2 drop Jedi Knight II

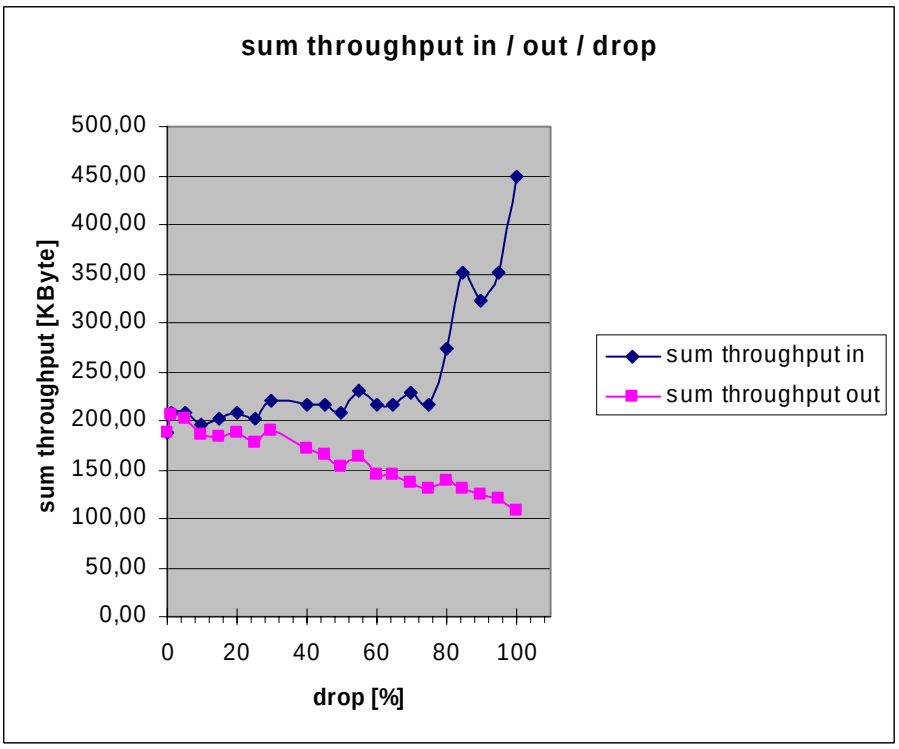

Abbildung 4.10: sum throughput in / out / drop jedi knight

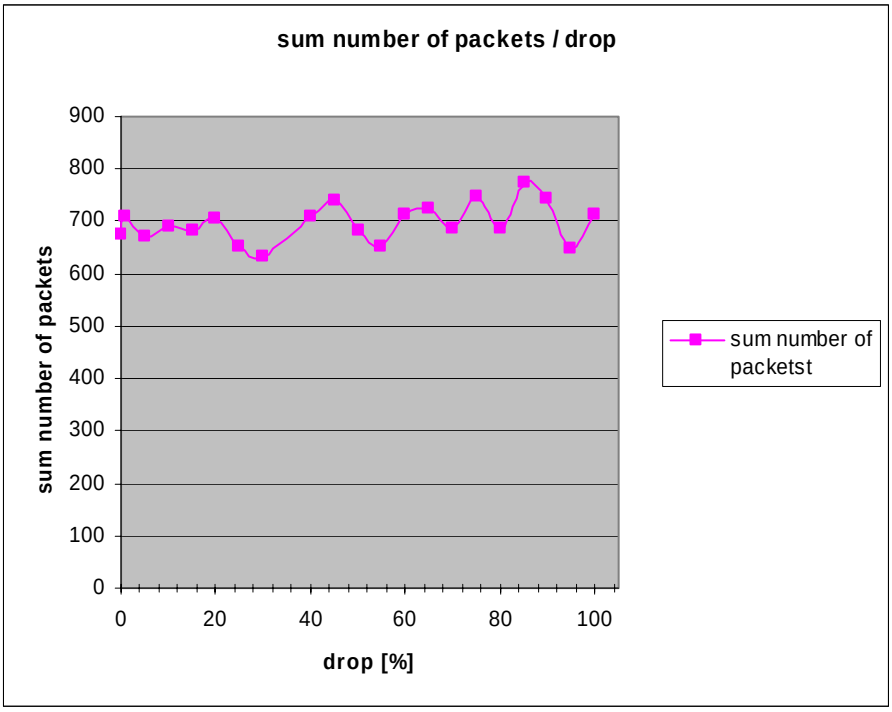

Abbildung 4.11: sum number of packets / drop jedi knight

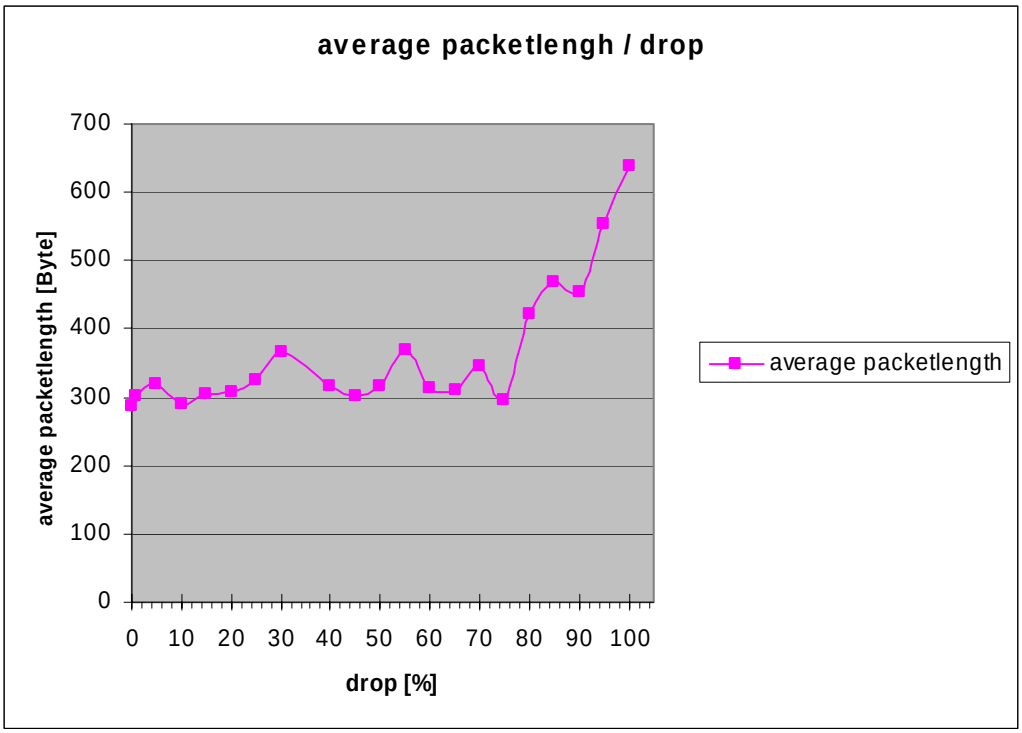

Abbildung 4.12: average packet length / drop jedi knight

Im Unterschied zu Fifa Football 2004 konnten beim Spiel Jedi Knight II die Versuche von 0 bis 100% Verlust durchgeführt werden, aber auch hier traten bei gewissen Verlustraten zwischenzeitliche Verbindungsunterbrechungen zum Client auf. Anschließend konnte jedoch das Spiel wieder fortgesetzt werden.

Die Summe der Pakete, die vom Server an den Client geschickt werden, wird bis zu einer Verlustrate von ca. 70% nur minimal erhöht (ca. 200KByte bis 220KByte). Wird die Verlustrate weiter erhöht, steigt der Durchsatz rapide an und erreicht bei ca. 450 KByte sein Maximum. Die empfangenen Pakete vom Client nehmen auf Grund der Tatsache, dass die gesendeten Pakete nur minimal ansteigen, mit Erhöhung der Verlustrate stetig ab (Abbildung 4.10). Die Summe der Anzahl der Pakete kann während des gesamten Versuchszeitraums im Mittel als konstant angenommen werden (Abbildung 4.11). Die durchschnittliche Paketlänge bleibt bis zu ca. 70% annähernd konstant und nimmt anschließend rapide zu (Abbildung 4.11).

In den Diagrammen (J.d.1%.1 bis J.d.80%.3) ist ersichtlich, dass bei höheren Verlustraten (im Zeitraum zwischen 30 und 90 Sekunden) der Durchsatz stark ansteigt (Diagr. J.d.80%.1). Die Pakete pro Sekunde (z.B. Diagr. J.d.50%.2 und Diagr. J.d.80%.2) bleiben wie die Summe der Pakete während des gesamten Versuchszeitraumes annähernd konstant. Die durchschnittliche Paketlänge pro Sekunde nimmt wie auch der Durchsatz bei höheren Verlustraten enorm zu (Diagr. J.d.80%.3).

4.3 iVisit

4.3.1 delay iVisit

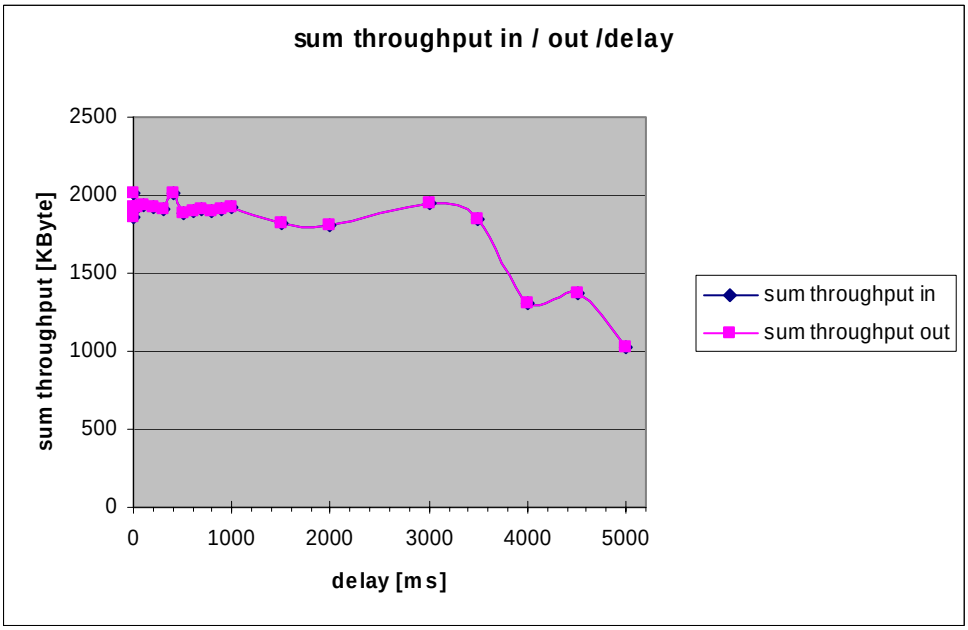

Abbildung 4.13: sum throughput in / out / delay iVisit

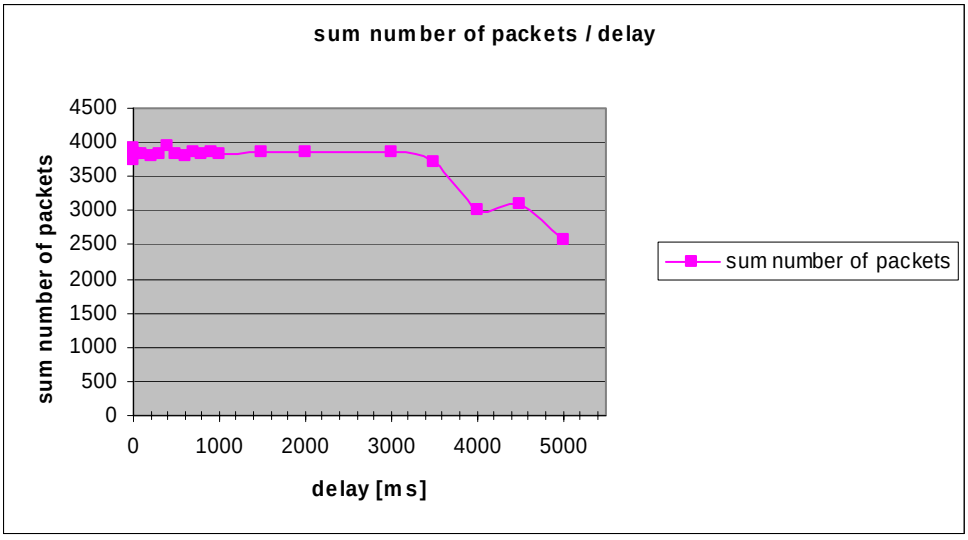

Abbildung 4.14: sum number of packets / delay iVisit

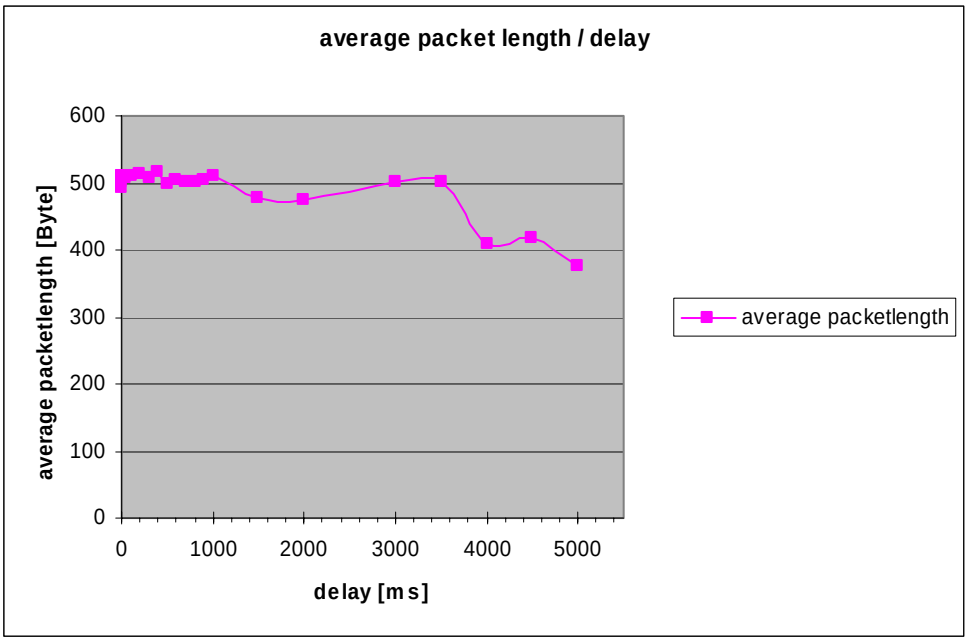

Abbildung 4.15: average packet length / delay iVisit

Bei allen Diagrammen (Abbildung 4.13 bis Abbildung 4.15) bleibt die Summe des Durchsatzes, die Summe der Anzahl der Pakete und die durchschnittliche Paketlänge bei iVisit bis zu einer Zeitverzögerung von ca. 3000ms beinahe konstant. Wird die Zeitverzögerung weiter erhöht, nehmen alle drei Werte stetig je nach Zeitverzögerung ab. Beim Videokonferenz Tool iVisit fällt auf, dass die Summe des Durchsatzes während des gesamten Versuchszeitraums im Vergleich zu den beiden Spielen um einiges höher ist.

Beim aufgezeichneten Durchsatz des Videokonferencing Tools iVisit kann bei einem größeren Zeitverzögerungswert beobachtet werden, dass zwar am Anfang der Zeitverzögerungsperiode (ab der 30. Sekunde) der Durchsatz relativ hoch ist, dieser aber anschließend kontinuierlich abnimmt. Das heißt, der Durchsatz reagiert direkt auf die größere Zeitverzögerung (Diagr. iV.d.2500ms.1 und Diagr. iV.d.5000ms.1). Auch bei den Paketen pro Sekunde und bei der durchschnittlichen Paketlänge pro Sekunde ist eine Abnahme bei größeren Zeitverzögerungen zu beobachten (Diagr. iV.d.5000ms.3 und Diagr. iV.d.5000ms.4)

4.3.2 drop iVisit

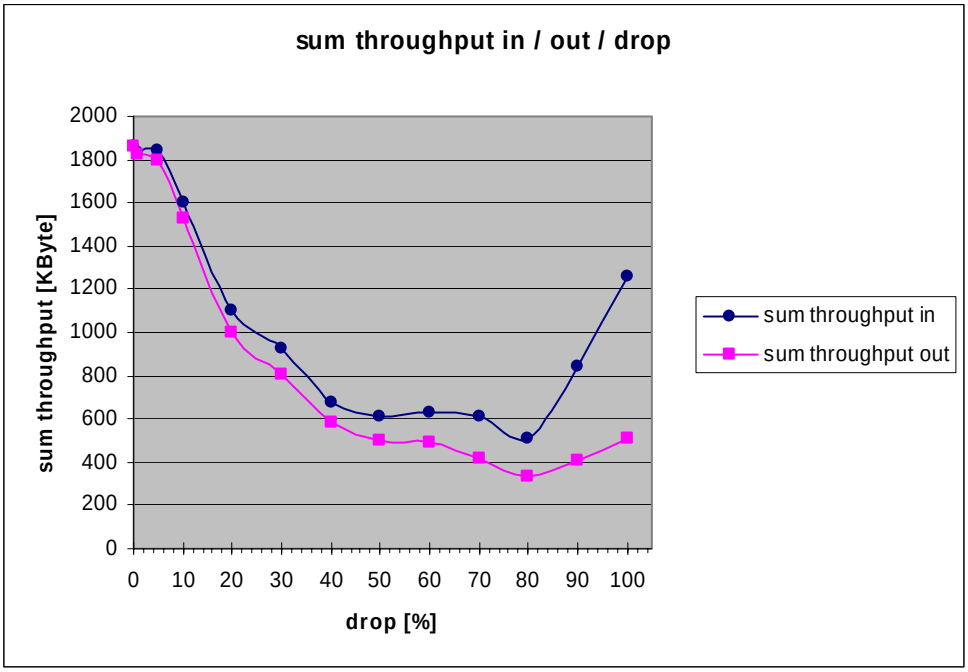

Abbildung 4.16: sum throughput in / out / drop iVisit

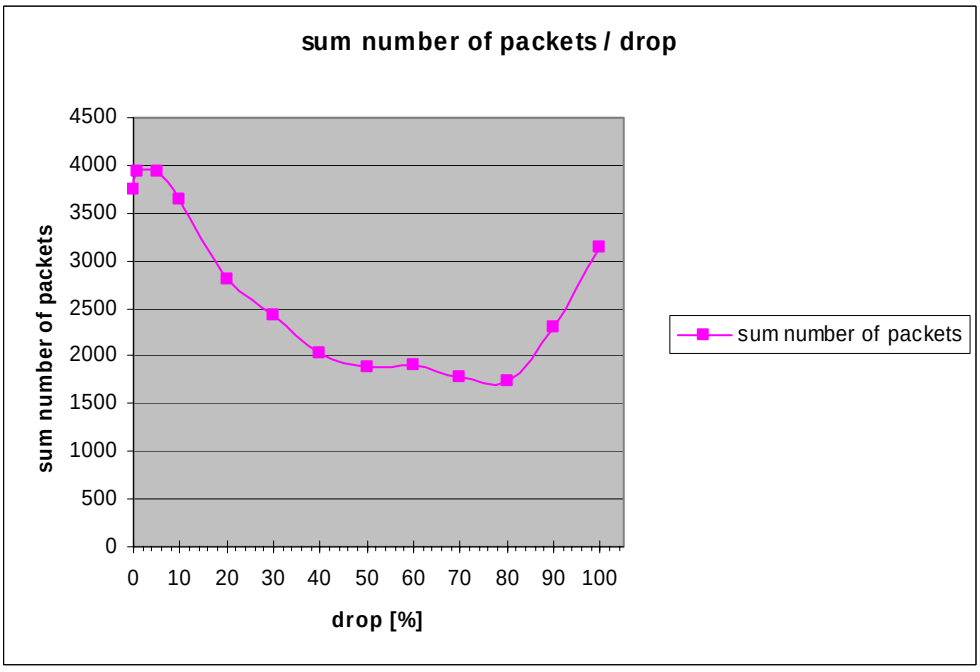

Abbildung 4.17: sum number of packets / drop iVisit

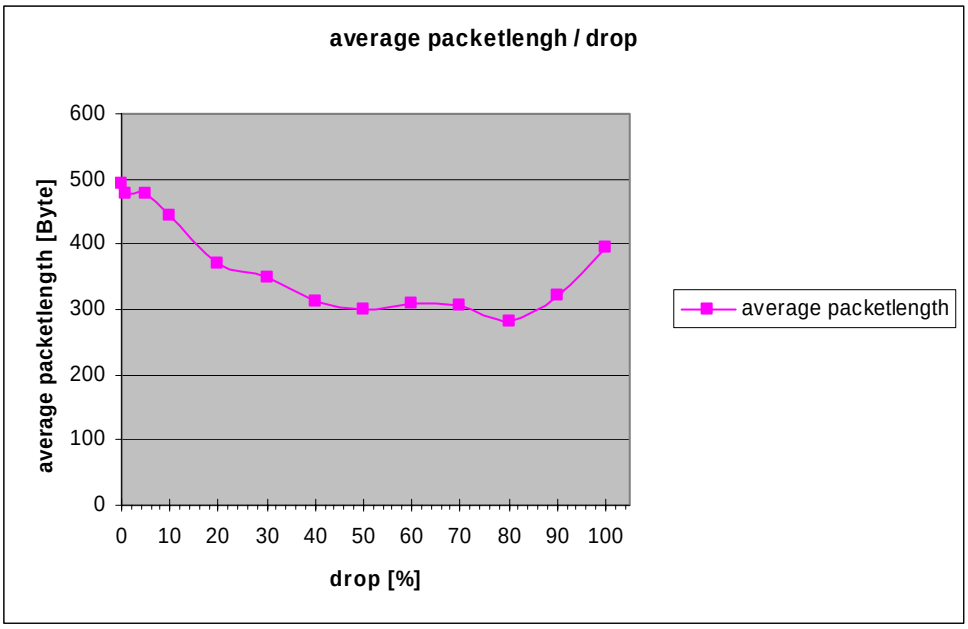

Abbildung 4.18: average packet length / drop iVisit

In Abbildung 4.16 (sum throughput) ist ersichtlich, dass die Summe der gesendeten und empfangenen Pakete ab einer Verlustrate von ca. 5% stark absinkt. Das bedeutet, dass iVisit mit einer größeren Verlustrate auch den Durchsatz verringert. Ab einer Verlustrate von ca. 80% wird aber die Anzahl der gesendeten Pakete des Servers wieder erhöht. Dieses Verhalten spiegelt sich auch in Abbildung 4.17 und Abbildung 4.18 wieder.

Auch Diagr. iV.d.50%.1 bis Diagr. iV.d.80%.3 zeigt, dass während des Zeitbereichs der Verlustemulation der Durchsatz, die Pakete pro Sekunde und die durchschnittliche Paketlänge pro Sekunde abnehmend sind. Bei Verlustraten über 80% hingegen kann iVisit erst stark verzögert auf die Paketverluste reagieren (Diagr. iV.d.90%.1 bis Diagr. iV.d.90%.3). Daraus resultiert auch der Anstieg der Summe des Durchsatzes, der Summe der Anzahl der Pakete und der durchschnittlichen Paketlänge (Abbildung 4.16 bis Abbildung 4.18).

5 Zusammenfassung

Mit der vorliegenden Arbeit sollte das Netzwerkverhalten von den Spielen Fifa Football 2004, Jedi Knight II und des Videokonferncing Tools iVisit mit Hilfe des Netzwerkemulators NistNet genauer analysiert werden. Dadurch mussten verschiedene „delay-Zeiten“ und „drop-Raten“ auf einem Router emuliert werden und anschließend aufgrund dieser Ergebnisse auf das Verhalten im realen Netzwerk geschlossen werden. Insbesondere wurden dabei die Diagramme „Summe Durchsatz“, „Summe Anzahl der Pakete“ und „durchschnittliche Paketlänge“ erstellt und analysiert. Im Anschluss folgt eine Zusammenfassung der Ergebnisse.

5.1 Fifa Football 2004

Bei Fifa Football 2004 wurde eine Emulation einer Zeitverzögerung von 0 bis 1950ms durchgeführt, da bei einer größeren Zeitverzögerung die Verbindung zum Client unterbrochen wurde und somit das Spiel nicht mehr fortgesetzt werden konnte. Im Wesentlichen konnte festgestellt werden, dass bei größeren Verzögerungszeiten der Durchsatz und die Anzahl der gesendeten Pakete abnimmt. Hingegen bleibt die die durchschnittliche Paketlänge eher konstant.

Bei Emulation von verschiedenen Verlustraten konnten die Versuche von 0 bis 70% ohne Verbindungsunterbrechung durchgeführt werden. Es stellte sich heraus, dass der Durchsatz, die Anzahl der Pakete und die durchschnittliche Paketlänge mit zunehmender Verlustrate steigend ist.

5.2 Jedi Kinght II

Bei Jedi Knight II wurde die Emulation einer Zeitverzögerung von 0 bis 5000ms durchgeführt. Bei diesem Spiel traten ab einer gewissen Zeitverzögerung zwar auch kurze Verbindungsunterbrechungen zum Client auf, aber das Spiel konnte anschließend fortgesetzt werden. Es konnte festgestellt werden, dass mit zunehmender Zeitverzögerung der Durchsatz und die durchschnittliche Paketlänge ansteigend ist. Die Anzahl der Pakete blieb jedoch während des gesamten Versuchzeitraums beinahe konstant.

Bei Emulation der verschiedenen Verlustraten konnten die Versuche von 0 bis 100% durchgeführt werden. Bei zu hohen Verlustraten war am Client nur mehr ein Standbild ersichtlich, aber das Spiel konnte anschließend wieder fortgesetzt werden. Auch hier zeigte sich, dass mit zunehmender Verlustrate der Durchsatz und die durchschnittliche Paketlänge ansteigt. Die Anzahl der Pakete blieb auch hier bei einer Verlustrate von 0 bis 100% relativ stabil.

5.3 iVisit

Bei iVisit wurde ebenfalls eine Zeitverzögerung von 0 bis 5000ms emuliert. Bei zu großem Wert der Zeitverzögerung war am Client nur mehr ein Standbild sichtbar. Bei diesem Videokonferencing Tool blieb bei Emulation einer Verzögerungszeit der Durchsatz, die Anzahl der Pakete und die durchschnittliche Paketlänge bis zu ca. 3000ms konstant. Weiters nahmen alle drei Werte bei Erhöhung der Zeitverzögerung stetig ab.

Die Verlustraten wurden ebenfalls von 0 bis 100% (zwischen 30 und 90 Sekunden) emuliert. Bei einer zu großen Verlustrate konnte ebenfalls nur ein Standbild am Client wahrgenommen werden. Hier wurde festgestellt, dass bis zu ca. einer Verlustrate von 80% der Durchsatz, die Anzahl der Pakete und die durchschnittliche Paketlänge absinkt aber dann bei weiterer Erhöhung der Verlustrate wieder ansteigt.

5.4 Danksagung

Für die Betreuung dieser Arbeit möchte ich mich besonders bei Herrn Dr.-Ing. Michael Welzl für seine tatkräftige Unterstützung bedanken. Weiters danke ich dem Institut für Informatik der Universität Innsbruck für die Bereitstellung der Teststrecke. Mein Dank gilt auch den Studenten Marcus Fischer und Andreas Radinger, deren Skripte mir eine große Hilfe beim Durchführen der Testabläufe waren.

5.5 Literaturverzeichnis

[1] Analyse des Netzwerkverhaltens von Echtzeit-Multimedia-Internetanwendungen detaillierte Analyse von Videoconferencing
http://www.welzl.at/teaching/baks/rolandwallnoefer/BAK_RolandWallnoefer.pdf

[2] Analyse des Netzwerkverhaltens von Computerspielen
http://www.welzl.at/teaching/baks/Bakarbeit_Computerspiele_Fischer_Radinger.pdf

[3] Enzyklopädie verwendet Erläuterung Emulation
http://de.wikipedia.org/wiki/Hauptseite

[4] Offizielle Home Page NistNet
http://www-x.antd.nist.gov/nistnet/

[5] Teststreckenaufbau
Akdag Muhlis – 29.04.2004
http://www.welzl.at/teaching/baks/akdagmuhlis/BakArbeitAkdagMuhlis.pdf

[6] Dokumentation von tcpdump
Programm zur Protokollierung des Netzwerkverkehrs
http://www.tcpdump.org/

[7] Dokumentation von mgen
Tool zum Erzeugen von zuvor definiertem Hintergrundverkehr
http://mgen.pf.itd.nrl.navy.mil/mgen.html

[8] Link Fifa Football 2004
http://www.easports.com/games/fifa2004/

[9] Link Jedi Knight II
http://www.lucasarts.com/products/outcast/html/