Leopold – Franzens - Universität
Innsbruck

Institut für Informatik

Genauere Analyse des Netzwerkverkehrs von Computerspielen

Bakkalaureatsarbeit

eingereicht bei Dr. Ing. Michael Welzl

Autor: Florian Larch und Markus Pabst

Innsbruck 08.11.2007

Inhaltsverzeichnis

1. Abkürzungsverzeichnis

Abkürzung	Bedeutung
ARP	Address Resolution Protocol
DNS	Domain Name System
DHCP	Dymanic Host Configuration Protocol
IP	Internet Protocol
LAN	Local Area Network
MAC	Medium Access Control
TCP	Transmission Control Protocol
OSI-ISO	Open Systems Interconnection Reference Model
UDP	User Data Protocol
Trpr	Tcpdump Rate Plot Real-time

2. Allgemeines

2.1. Einleitung

Computerspiele (Single- Multiplayer) haben sich in den letzten 20 Jahren sehr stark entwickelt. In letzter Zeit haben besonders Multiplayercomputerspiele immer mehr an Popularität gewonnen. Vor allem Online-Spiele liegen voll im Trend. Online-Spiele sind Spiele, in denen mehrere tausend User via Internet in ein und derselben Spielwelt miteinander bzw. gegeneinander spielen können. In diesen simulierten Szenarien ist es notwendig viele Daten untereinander auszutauschen. Der dabei entstehende Verkehr stellt hohe Anforderungen an Software und Hardware. Um den entstehenden und immer wachsenden Datenfluss zu bewältigen, müssen diese ständig verbessert werden.

Damit ein Spiel von möglichst vielen Spielern genutzt werden kann, muss es unter sehr geringen technischen Voraussetzungen einwandfrei funktionieren. Um die Netzperformance zu erhöhen, integriert man in Spiele den sogenannten Netcode, welcher für die Datenübertragung im Netzwerk verantwortlich ist. Ein moderner Netcode besitzt verschiedene Strategien, um in einer Stausituation entsprechend dem Datenfluss zu reagieren.

In dieser Bakkalaureatsarbeit wird untersucht, welche Strategien es gibt.

2.2. Begriffserklärung

Im folgenden Kapitel wird die zugrunde liegende Technologie bei Computerspielen genauer erklärt.

2.2.1. Client-Server Technologie

Die analysierten Computerspiele basieren auf einer Client-Server Architektur.

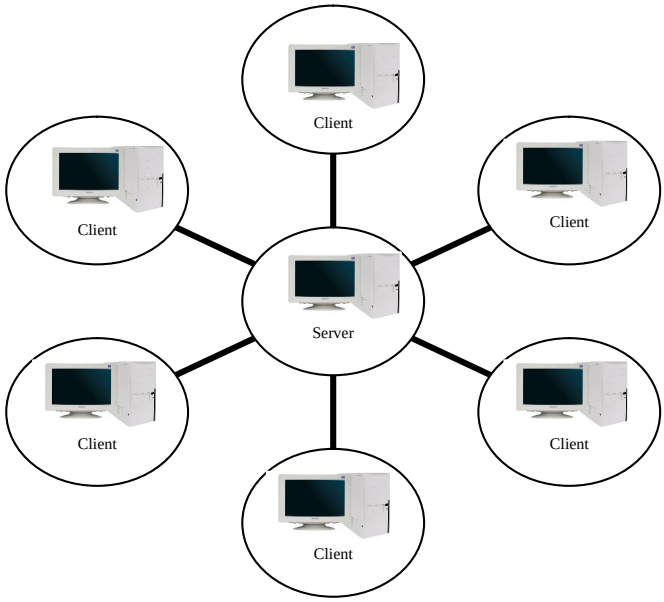


Abbildung 1: Client-Server Architektur

Der aufgrund seiner Struktur auch „Stern-Architektur“ genannte Aufbau, siehe Abb. 1, enthält einen Server, welcher mit mehreren Clients verbunden ist. Der Server speichert sämtliche Nachrichten, die er von den angehängten Clients bekommt, verarbeitet diese und schickt sie an die Clients zurück. Nur so können die Spielstände für alle Clients synchronisiert werden.

Der Vorteil dieses Systems ist, dass der Server als reine Rechenanlage keine grafischen Darstellungen des Spielgeschehens benötigt und daher viel Rechenzeit spart. Die Clients dagegen haben einen hohen Rechenaufwand, der durch das Rendern der verschiedenen Szenarien entsteht.

Allerdings hat die Stern-Architektur auch Nachteile. Sind bei einem Server zu viele Clients eingeloggt, kann es zu Verzögerungen oder sogar zur Trennung der Verbindung kommen (Time Out). Auch ein synchrones Ablaufen der Spiele kann in diesem Fall nicht mehr gewährleistet werden. Das kann auch damit zusammenhängen, dass die Verbindungen zwischen Server und Clients unterschiedlicher Qualität sind. (unterschiedliche Hardware der Clients & unterschiedliche Qualität der Datenübertragung).
Trotzdem gehört die Stern-Architektur derzeit zu den effizientesten Lösungen für vernetztes Spielen und wird daher von vielen Spielherstellern verwendet.

2.2.2 Game-Engine

Als Game-Engine wird der eigenständige Teil des Computerspiels bezeichnet. Dieser Teil beinhaltet alle nötigen Routinen, Objekte und Bewegungen das Spiel zu berechnen und auf dem Bildschirm darzustellen. Eine Game-Engine ist eine Art Programmbibliothek, die den Entwicklern von Computerspielen häufig benutzte Werkzeuge zur Verfügung stellt.

2.2.3 Netcode

Der Netcode ist jener Teil der Game Engine, der die Aufgabe besitzt, sämtliche Spielinformationen im Netzwerk zu koordinieren. Dort werden Objekte oder auch Bewegungen und andere benötigte Spielinformationen eines Netzwerkspiels in Informationspakete zusammengefasst. Diese werden mittels des Netcodes vom Client zum Server oder vom Server zum Client geschickt. Dadurch wird ein Zusammenspiel mehrerer Clients möglich. Zudem bestimmt der Netcode die Voraussetzungen für den Multiplayerteil eines Spiels, z. B. wie viele Spieler gleichzeitig am Spiel teilnehmen können oder ob ein 56k-Modem ausreichend ist oder DSL benötigt wird.

3. Aufgabenstellung

Ziel dieser Arbeit war die genauere Untersuchung des Netzwerkverhaltens von Computerspielen bei Verzögerung oder Verlust von Paketen. Dazu wurde ein kleines Netzwerk mit fünf Computern eingesetzt werden. Ein Computer wird als Linux-Router verwendet, welcher das Netz in zwei Teile gliedert. Zwei Computer bilden ein IP-Netz und die restlichen zwei Computer bilden ein weiteres IP–Netz. Auf jeweils einem Computer pro Netz wird ein Computerspiel gestartet. Mit den anderen zwei Computern wird der auftretende Netzwerkverkehr aufgezeichnet. Dann wird am Router wird eine Verzögerung oder ein prozentueller Verlust von Paketen simuliert.

Diese Arbeit beinhaltet folgende Punkte:

- Vermittlung der Netzwerkgrundkenntnisse
- Vermittlung einiger Grundkenntnisse der beiden Betriebssysteme Windows und Linux
- Installation von allen notwendigen Tools und Computerspielen
- Erstellen von normierten Testfällen
- Aufzeichnen des auftretenden Netzwerkverkehrs
- Analyse des auftretenden Netzwerkverkehrs mit graphischer Darstellung
- Interpretation der Testergebnisse

Folgenden Fragen werden in dieser Arbeit beantwortet:

- Wie reagieren Computerspiele auf Verlust bzw. Verzögerung von Paketen bei der Datenübertragung?
- Welche Strategie verwenden Computerspiele, um die Kommunikation zu gewährleisten?

4. Netzwerkprotokolle

Basisinformationen zum UDP Protkoll:

<table>
<tr><td colspan="2">UDP (User Datagram Protocol)</td></tr>
<tr><td>Familie:</td><td>Internetprotokollfamilie</td></tr>
<tr><td>Einsatzgebiet:</td><td>Verbindungslose Übertragung
von Daten über das Internet</td></tr>
<tr><td colspan="2">UDP im TCP/IP-Protokollstapel
<table>
<tr><td>Anwendung</td><td>DNS</td><td>DHCP</td><td>NTP</td><td>…</td></tr>
<tr><td>Transport</td><td colspan="2">UDP</td><td colspan="2">TCP</td></tr>
<tr><td>Internet</td><td colspan="4">IP</td></tr>
<tr><td>Netzzugang</td><td>Ethernet</td><td>Token Ring</td><td>FDDI</td><td>…</td></tr>
</table></td></tr>
<tr><td>Standards:</td><td>RFC 768 (1980)</td></tr>
</table>

Abbildung 2: UDP (User Datagram Protocol)

4.1. IP

Das „Internet Protocol“ (IP) ist ein, in Computernetzen, weit verbreitetes Netzwerkprotokoll. Es ist die Implementierung der Internet-Schicht des TCP/IP-Modells bzw. der Vermittlungsschicht (Network Layer) des OSI-Modells.

IP bildet die vom Übertragungsmedium unabhängige Schicht der Internetprotokollfamilie. Das bedeutet, dass mittels IP-Adresse und Subnetzmaske Computer innerhalb eines Netzwerkes in logische Einheiten, so genannte Subnetze, gruppiert werden können. Auf dieser Basis ist es möglich, Computer in größeren Netzwerken zu adressieren und so Verbindungen zu ihnen aufzubauen. Denn logische Adressierung ist die Grundlage für Routing. Das Internet Protokoll stellt die Grundlage des Internets dar.

4.2. UDP

Das User Datagram Protocol (Abk. UDP) ist ein einfaches, verbindungsloses Netzprotokoll, das zur Transportschicht der Internetprotokollfamilie gehört. Die Aufgabe von UDP ist es, Daten, die über das Internet übertragen werden, der richtigen Anwendung zukommen zu lassen.

Die Entwicklung von UDP begann 1977, als man für die Übertragung von Sprache ein einfacheres Protokoll benötigte als das bisher verwendete, verbindungsorientierte TCP. Es wurde ein Protokoll benötigt, das ausschließlich für die Adressierung zuständig war, ohne die Datenübertragung zu sichern. Nur so konnte Verzögerungen bei der Sprachübertragung vorgebeugt werden.

Funktionsweise:

Um die Daten, welche via UDP versendet werden, dem richtigen Programm auf dem Zielrechner zukommen zu lassen, werden bei UDP so genannte Ports verwendet. Dazu wird bei UDP die Portnummer des Dienstes mit gesendet, der die Daten erhalten soll. Diese Erweiterung der Host- zu Host- auf eine Prozess- zu Prozessübertragung wird als Anwendungsmultiplexen bzw. Anwendungsdemultiplexen bezeichnet.
Zusätzlich bietet UDP die Möglichkeit einer Integritätsüberprüfung an, indem eine Prüfsumme mit gesendet wird. Dadurch kann eine eventuell fehlerhafte Übertragung erkannt werden.

5. Allgemeine Basisinformationen

5.1. Qualität des Netzwerks

5.1.1. Latenz

Latenz ist der allgemeine Begriff für die Zeitdifferenz zwischen dem Auslösen eines Ereignisses und der Ausführung der zugehörigen Aktion. Werden die Ereignisse als Netzwerknachrichten verschickt, so berechnet sich die Gesamtlatenz als Summe der Einzellatenzen der folgenden Prozesse:

Detektion: Die Auslösung eines Ereignisses wird erkannt, z.B. wird eine Kollision registriert.

Senderseitige Verarbeitung: Das Eingabeereignis wird auf dem Computer, auf dem das Ereignis ausgelöst wurde, instanziiert und verarbeitet.

Serialisierung: Das Ereignis wird in binäre Form gebracht und als Netzwerknachricht verpackt.

Übertragung: Die Nachricht wird über das Netzwerk vom Sender zum Empfänger transportiert.

Deserialisierung: Das Ereignis wird auf der Empfängerseite aus der Netzwerknachricht wiederhergestellt.

Empfänger-Verarbeitung: Das Ereignis wird auf der Empfängerseite weiterverarbeitet.

Die minimale Übertragungszeit einer Information wird durch di maximale Übertragungsgeschwindigkeit definiert. So hat beispielsweise ein Glasfaserkabel eine höhere Übertragungsgeschwindigkeit (etwa 50 %) als ein Kupferkabel. Weitere Latenzen entstehen durch Verstärker und Repeater die in regelmäßigen vom Übertragungsmedium abhängigen Abständen das Signal verstärken. Auch Internet-Router bearbeiten jedes Paket und erhöhen dadurch die Latenzzeit. Besonders stark fallen Signalwandlungen, beispielsweise analog/digital in einem Modem, sowie Zwischenspeicherung von Informationen ins Gewicht.

5.1.2. Bandbreite und Verstopfung

Wird über eine Datenleitung mehr gesendet als die maximale Bandbreite zulässt, tritt eine Art Verstopfung (congestion) auf. Wenn ein Netzwerkrouter überlastet ist, nimmt er keine weiteren Nachrichten an, und oft gehen dadurch Nachrichten verloren. Es gibt verschiedene Strategien um Stauungen zu verhindern. Auf diese werden wir noch später genauer eingehen.

5.1.3. Verlässlichkeit

Durch Ausfälle und Störungen auf dem Übertragungsweg kann eine Nachricht verfälscht werden oder gar verloren gehen. Viele Protokolle besitzen daher die Möglichkeit, defekte oder verloren gegangene Teile einer Nachricht zu detektieren oder gar kleinere Fehler zu korrigieren. Zur Erkennung verfälschter Nachrichten dienen beispielsweise Hashwerte, Message Authentication Codes oder digitale Signaturen. Alle Detektionsmechanismen erhöhen das Kommunikationsvolumen, da zusätzliche Informationen übertragen werden müssen. Auch die Übertragungslatenz steigt an, denn auf beiden Seiten ist eine zusätzliche Bearbeitung jeder Nachricht notwendig. Kleinere Übertragungsfehler können durch Vorwärts-Fehlerkorrektur (Forward Error Correction) auf der Empfängerseite korrigiert werden. Dabei werden in jeder Nachricht redundante Informationen mit gesendet, aus denen durch Verwendung festgelegter Algorithmen falsche Bits aufgespürt und korrigiert werden können. Nicht wiederherstellbare oder als verloren detektierte Nachrichten kann der Empfänger vom Sender neu anfordern.

5.1.4. Paket- und verbindungsbasierte Kommunikation

Kommunikationsverbindungen können paketbasiert (ZB UDP / IP) oder verbindungs-basiert (TCP / IP) sein. Paketbasierte Verbindungen übermitteln einzelne Pakete, auch Datagramme genannt (vergleichbar mit der Post). Verbindungsbasierte Verbindungen etablieren eine feste Verbindung zwischen den Kommunizierenden, auf der Daten in einem Datenstrom ausgetauscht werden können. In der paketbasierten Kommunikation ist weiters eine maximale Paketgröße (Maximum Transfer Unit) (MTU) zu beachten.

Wird sie überschritten, müssen Nachrichten fragmentiert, also in mehrere Pakete aufgeteilt werden. Fragmentierung kann dazu führen, dass Teilstücke alter Nachrichten den Eingangspuffer einer Verbindung blockieren und erst nach einem Timeout entfernt werden. Ist die Verbindung unzuverlässig, können verloren gegangene Fragmente nicht neu angefordert werden und bewirken den Verlust der gesamten Nachricht.

5.2. Verringerung der Netzwerklast

Es gibt verschiedene Methoden, welche die Netzwerklast verringern. Dadurch können Skalierbarkeit und Geschwindigkeit in einem Netzwerk erhöht werden.

5.2.1. Interessensbereiche und Gebietszerlegung

Eine Jeder-mit-Jedem-Kommunikation zwischen mehreren hundert Teilnehmern ist in Echtzeit nicht möglich. Zum Glück ist aber auch nicht jeder Teilnehmer an allen Nachrichten interessiert. In der Regel genügt es, wenn er nur den für ihn relevanten Teil der Nachrichten empfängt und verarbeitet. In einer auf mehrere Computer verteilten, virtuellen Welt sind das die Updates der Entitäten, die sich im Blickfeld der Kamera befinden. Diesen Bereich nennt man den Interessensbereich des Teilnehmers. Ein derartiges Filterungssystem heißt Area Of Interest Management (AOIM).
Die Berechnung sich ständig ändernder Interessensbereiche kann sehr schnell aufwändig werden kann. Deshalb zerlegt man die gesamte Szene, in Teilszenen, welche voneinander unabhängig verarbeitet werden können. Dieser Vorgang wird als Gebietszerlegung bezeichnet. Jeder Teilnehmer bekommt dann alle Nachrichten aus dem Teilgebiet, in dem sich seine Entität gerade befindet, und eventuell noch Informationen aus den direkt benachbarten Bereichen.
Bei der Zerlegung in Teilszenen lassen sich die Teile mit ähnlichem Inhalt in disjunkten Gebieten zusammen fassen z.B. in verschiedene Gebäude, Etagen oder Räume.

5.2.2. Verhaltensmuster

Große Zustandsänderungen äußern sich oft in einer Vielzahl vorhersehbarer kleiner Änderungen. Geht beispielsweise ein humanoider Avatar von einem Ort zu einem anderen, so bewegen sich seine Arme und Beine nach einem deterministischen Muster. Kennt ein Teilnehmer dieses Muster, kann aus der Nachricht „Avatar geht von A nach B“ dort lokal das komplette Bewegungsmuster rekonstruiert werden, das aus einer Vielzahl von Teilereignissen besteht. Eine solche Zusammenfassung von Ereignissen nennt man Verhaltensmuster (behaviour). Typische Verhaltensmuster für Avatare sind beispielsweise „gehen“, „laufen“, „stehen“ oder „angreifen“. Durch Verhaltensmuster kann die Anzahl der zu übertragenden Ereignisse erheblich reduziert werden. Allerdings steigt dabei der Rechenaufwand für die Teilnehmer.

6. Versuchsdurchführung

6.1. Allgemeines zu Teststrecke

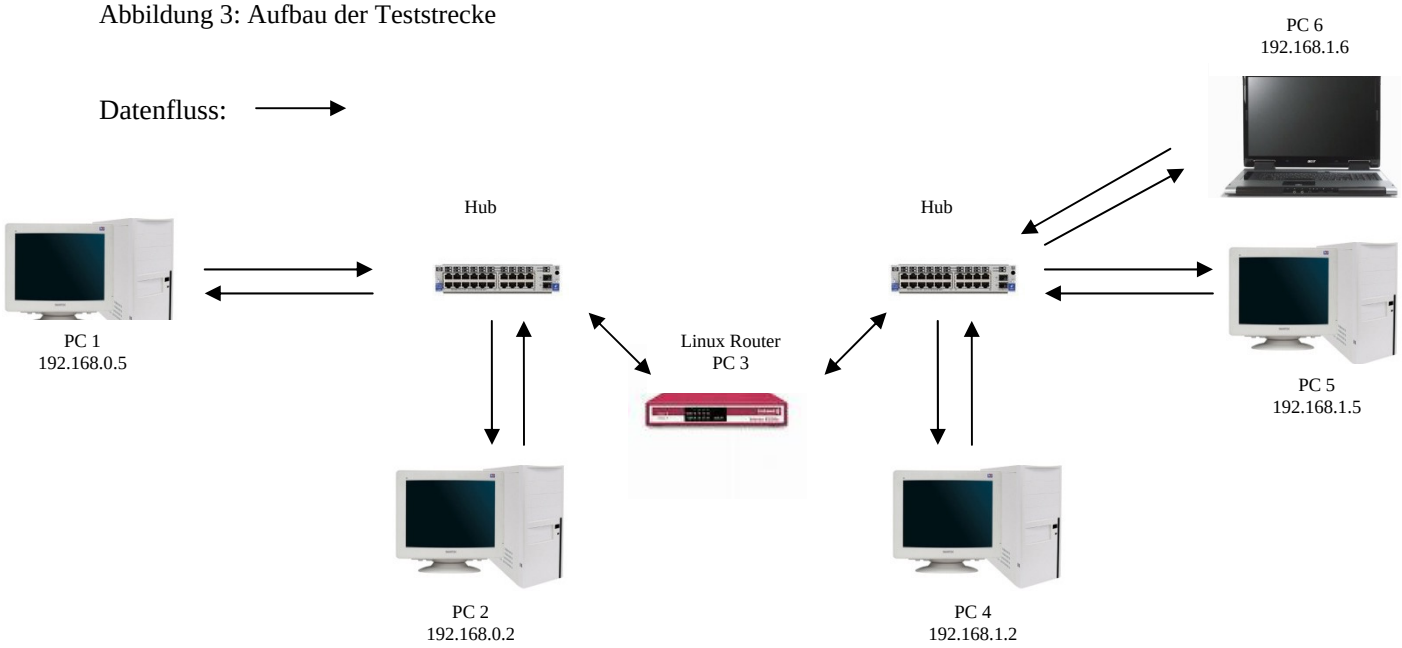
Die Teststrecke bestehend aus fünf Rechnern ist wie folgt zusammengesetzt:

- 2 Windowsrechner (PC1, PC5 bzw. PC6)
- 2 Linuxrechner (PC2, PC4)
- 1 Router (PC 3)

Im Versuchsaufbau wird ein Windowsrechner (PC5 oder PC6) als Server eingesetzt. Dieser ist über einen Hub mit einem Linuxrechner, der als Messstation fungiert und mit dem Router verbunden ist. Der 1. Windowsrechner (PC5 oder PC6) simuliert einen Spielserver, der 2. Windowsrechner (PC1) einen Client, auf dem die Anwendungen gestartet werden (Siehe Abbildung 3).
Der Netzwerkverkehr dieser Anwendungen wird mit Hilfe der 2 Linuxrechner (PC2, PC4) analysiert und ausgewertet. Die Linuxrechner (PC2, PC4) messen den Datenfluss zwischen Server und Client.
Die Aufgabe des Routers ist es eine Verzögerung bis maximal 3 Sekunden und den Verlust von maximal 90 % der Pakete zu simulieren. Der Datenfluss zwischen Server und Client wird durch die Linuxrechner mit protokolliert.

6.2. Aufbau der Teststrecke

Abbildung 3: Aufbau der Teststrecke



6.3. Ablauf des Testverfahrens

Bevor mit dem Testen begonnen werden kann, müssen alle nötigen Anwendungen auf den jeweiligen PCs installiert und konfiguriert werden.

1. Auf beiden Windows XP Rechnern werden die jeweiligen Spiele gestartet. PC5 (Server) richtet ein Netzwerkspiel ein und PC1 (Client) baut eine Verbindung zum Server auf.

2. Auf PC2 wird der Netzwerksniffer Wireshark gestartet, welcher den Netzwerkverkehr von eth1 protokolliert.

3. Auf PC4 wird der Netzwerksniffer Wireshark gestartet, welcher den Netzwerkverkehr von eth1 protokolliert.

4. Auf dem Router (PC3) wird das Shell –Skript testszenario.sh[1] ausgeführt und somit eine nistnet - Konfiguration aktiviert.

5. Nach 120 Sekunden können alle Anwendungen gestoppt werden.

Auf PC2 werden die beiden dump-Files von Wireshark gespeichert und stehen später für die Auswertung zur Verfügung. Das dump-File auf PC4 hat den Netzwerkverkehr vor dem Router mit protokolliert und das dump-File auf PC2 hat den Netzwerkverkehr nach dem Router mit protokolliert.

6.4. Testszenarien

Im Vorfeld wurden die Testszenarien normiert.

Testszenario Verzögerung (delay):	Testszenario Verlust (drop):
ms → Millisekunden	
1. **Szenario**: 100 ms 2. **Szenario**: 200 ms 3. **Szenario**: 300 ms 4. **Szenario**: 500 ms 5. **Szenario**: 1000 ms 6. **Szenario**: 1500 ms 7. **Szenario**: 2000 ms 8. **Szenario**: 3000 ms	1. **Szenario**: 10 % 2. **Szenario**: 20 % 3. **Szenario**: 30 % 4. **Szenario**: 40 % 5. **Szenario**: 50 % 6. **Szenario**: 60 % 7. **Szenario**: 70 % 8. **Szenario**: 80 % 9. **Szenario**: 90 %

Dabei wurde nicht unterschieden in welcher Spielsituation sich das Spiel gerade befand, weil davon ausgegangen wurde, dass über die Dauer des Tests nicht nur eine einzelne Spielsituation auftreten würde und das Testergebnis somit keiner speziellen Spielsituation zugeordnet werden kann.

[1] Dateiname: stet für jeweiligen Testfall z. B drop10.sh , delay100.sh, …

6.5. Spezifikation der Teststrecke

6.5.1. Hardwarespezifikation

Der Testablauf wurde auf den hier aufgelisteten Personalcomputern bzw. auf einem Laptop durchgeführt. Diese Rechner hatten folgende Eigenschaften:

PC1 (Windowsrechner links):

Prozessor: Intel Pentium 4 1,5 GHz
Grafikkarte: Nvidia GeForce 6600 AGP 4x 256MB
RAM: 1024 MByte
Netzwerkkarte: Realtek RTL 8139 – Familie - PCI – Fast Ethernet - NIC

PC2 (Linuxrechner links):

Prozessor: Intel Pentium 4 1,8 GHz
Grafikkarte: Video Cart Intel Cooperation 82845G/GL/GE Integrated Device 128 MByte
RAM: 1024 MByte
Netzwerkkarte: Intel Cooperation 82540EM

PC3 (Router):

Prozessor: AMD Athlon(TM) XP 2200+
Grafikkarte: ATI Radeon 7000 64 MB AGP 4x
RAM: 512 MB
Netzwerkkarte:
eth1:
Realtek Semiconductor co.,ltd.RTl-8139/8139c/8139c+
eth2:
Realtek Semiconductor co.,ltd.RTl-8139/8139c/8139c+

PC4 (Linuxrechner rechts):

Prozessor: Intel Pentium 4 1,7 GHz
Grafikkarte: Nvidia GeForce 2Gts Pro 128 MByte
RAM: 512 Mbyte
Netzwerkkarte: 3Com 3c906C

PC5 (Windowsrechner rechts):

Prozessor:	Intel Pentium 4 1,5 GHz
Grafikkarte:	Nvidia GeForce 3 Ti 300 64MB
RAM:	1024 MByte
Netzwerkkarte:	3Com EtherLink XL PCI

PC6 (Windowsrechner):

Prozessor:	Intel Core 2 T5500 1,66 GHz
Grafikkarte:	NVidia GeForce Go 7300 256 MB
RAM:	2 GByte
Netzwerkkarte:	Broadcom 440 10/100Integrated Controller

6.5.2. Softwarespezifikation

Die Rechner in der Teststrecke hatten folgende Software installiert:

PC1:	**Betriebssystem**:	Microsoft Windows Professional SP2
	Verwendungszweck:	Rechner, auf dem die Spiele installiert sind und laufen.
PC2:	**Betriebssystem**:	Linux Fedora Core 5 **Kernel**: 2.6.18
	Verwendungszweck:	Aufzeichnung des Netzwerkverkehrs
PC3:	**Betriebssystem**:	Linux Fedora Core 4 **Kernel**: 2.6.15
	Verwendungszweck:	Router (Bottleneck)
PC4:	**Betriebssystem**:	Linux Fedora Core 5 **Kernel**: 2.6.18
	Verwendungszweck:	Aufzeichnungen des Netzwerkverkehrs und Erzeugung des Hintergrundverkehrs
PC5:	**Betriebssystem**:	Microsoft Windows Professional SP2
	Verwendungszweck:	Rechner, auf dem die Spiele installiert sind und laufen.

PC6:	**Betriebssystem**:	Microsoft Windows Professional SP2
	Verwendungszweck:	Rechner, auf dem die Spiele installiert sind und laufen.

Anmerkung: PC6 wurde zum Testen der Spiele Quake4 und Battlefield 2 eingesetzt, da PC5 die Hardwarespezifikationen nicht erfüllte.

6.6. Tools für die Durchführung der Tests

6.6.1. Netzwerkspezifische Tools

Zur Analyse des Netzwerkes verwendeten wir diverse Tools. Mit Hilfe derer es möglich war den Datenfluss zwischen den Rechnern auszuwerten.

- **Wireshark**

Name: Wireshark
Verwendungszweck: Protokollierung des Netzwerkverkehrs
Firma: Open Source
Quelle: http://www.wireshark.org/
Beschreibung:
Wireshark (zu Deutsch „Kabelhai“ oder „Draht-Hai“, ehemals Ethereal genannt) ist ein so genannter Netzwerksniffer, welcher den Netzwerkverkehr mitprotokolliert, und für das menschliche Auge lesbar darstellt. Dabei stellt Wireshark die Datenpakete so genau dar, dass man auch gezielte Einblicke in die verschiedenen Protokollheader bekommen kann. Netzwerksniffer werden verwendet, um den Netzwerkverkehr aufzuzeichnen und zu analysieren.
Als Gerald Combs, Entwickler von Ethereal, von Ethereal Software Inc. zu CACE Technologies wechselte, startete er ein eigenes Folgeprojekt und nannte es Wireshark.

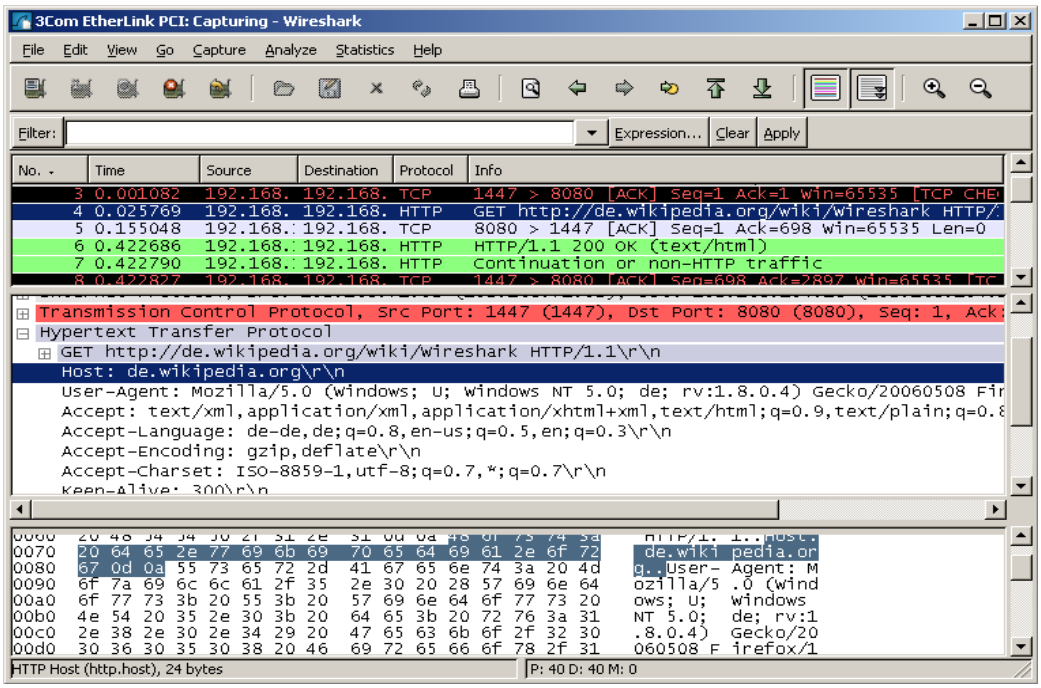


Abbildung 4: Screenshot des Programms Wireshark

- **Nistnet**

Name: Nistnet

Verwendungszweck: Netzwerlemulator

Firma: Open Source

Quelle: http://nistnet.com

Beschreibung: NISTNet ist ein Netzwerkemulator, der es ermöglicht die Eigenschaften einer Kommunikationsbeziehung hinsichtlich Verzögerung, Paketverlust, Duplikaten und Bandbreite zu beeinflussen. Version 3.0a ist die derzeit aktuelle Version für Kernel 2.6. Für Kernel 2.0 bis 2.4 muss Version 2.0.12b verwendet werden.

- **mgen**

Name: mgen

Verwendungszweck: Erzeugen von zuvor definiertem Hintergrundverkehr

Firma: Naval Research Laboratory

Quelle: http://mgen.pf.itd.nrl.navy.mil/mgen.html

Beschreibung: „mgen“ ist ein so genannter „trafficgenerator“. Das Programm ist Open Source und erzeugt einen definierbaren UDP Netzwerkverkehr zu einem frei wählbaren Zielrechner im Netzwerk. Dieses Tool wurde bei den verschiedenen Testfällen verwendet um den Hintergrundverkehr zu erzeugen. Das Programm benötigt ein Skriptfile, auf welches später noch genauer eingegangen wird.

6.6.2. Weitere Tools

- **WinPcap**

Name: WinPcap

Verwendungszweck: für WinDump nötig um auf Schnittstellen zugreifen zu können

Firma: Open Source

Quelle: http:// www.winpcap.org

Beschreibung:

WinPcap ist eine als Programmbibliothek, nutzbar als Open Source.

WinPcap besteht aus einem Treiber, welcher hardwarenahen Zugriff auf die Netzwerkkarte für Windows-basierte Betriebssysteme ermöglicht und auf eine Sammlung von Programmen, die den bequemen Zugriff auf die einzelnen für Netzwerke relevanten Schichten des OSI-Modells bieten. Die Programmbibliothek basiert auf der von Unix her bekannten Bibliothek „libpcap“.

Die über das Netzwerk transportierten Pakete werden durch die WinPcap-Module unter Umgehung des Protokollstacks entgegengenommen und weitergeleitet. Somit können Statistiken über die Netzwerkauslastung, die verschiedenen Paketarten und deren Inhalt protokolliert und analysiert werden. Konkrete Anwendung findet WinPcap in Netzwerküberwachungssoftware wie zum Beispiel Wireshark, Nmap, Snort, Cain & Abel, WinDump und ntop. In dieser Arbeit wurde die Auswertung auf einem Windowsrechner durchgeführt. Damit WinDump richtig arbeiten kann benötigt man WinPcap.

- **WinDump**

Name: WinDump

Verwendungszweck: Für Auswertung nötig

Firma: Dieses Programm ist Freeware. Es darf sowohl privat als auch kommerziell kostenlos eingesetzt werden.

Quelle: http://www.tcpdump.org/

Beschreibung:

Tcpdump/WinDump ist die bekannteste Software zur Überwachung und Auswertung von Netzwerkverkehr. Ursprünglich von Van Jacobson, Craig Leres und Steven McCanne geschrieben, wurde sie mittlerweile von vielen anderen weiterentwickelt. Tcpdump/WinDump arbeitet im Textmodus und wird über die Kommandozeile gesteuert.

Das Programm liest Daten, die in Form von Paketen über das Netzwerk gesendet werden und stellt diese auf dem Bildschirm dar oder speichert sie als Dateien. Zusätzlich ermöglicht Tcpdump/WinDump die Auswertung von vorher in Dateien gespeicherten Paketen. Mittels Parametern, die bei Programmstart auf der Kommandozeile angegeben werden müssen, steuert der Benutzer das Verhalten von WinDump und übergibt Filter an das Programm, wodurch die Pakete ausgewertet werden.

- **Trpr (Tcpdump Rate Plot Real-time)**

Name: Tcpdump Rate Plot Real-time

Verwendungszweck: Erzeugen von Plotfiles

Quelle: http://pf.itd.nrl.navy.mil/protools/trpr.html

Beschreibung

Trpr ist ein Auswertungsprogramm, das aus tcpdump/WinDump Output graphische Darstellungen verschiedener Flows erzeugt. Als Plotter wird extern auf das Programm Gnuplot zurückgegriffen.

- **Microsoft Office 2003 SP2**

- **Name:** Microsoft Office 2003 SP2

Verwendungszweck:

Word: Dokumentation dieser Arbeit

Excel: Auswertung der Paketgröße und der Paketanzahl

- **Adobe Acrobat Writer 7.0**

Verwendungszweck: Konvertierung der Dokumentation in PDF

7. Anwendungen

Für unsere Versuchsreihe verwendeten wir eine Reihe von Spielen aus verschiedenen Kategorien:

- Strategiespiele:
  - Age of Empires 3
  - Anno 1701
  - Dawn of War
- Shooterspiele
  - Battlefield 2
  - Quake 4
- Sportspiele
  - Fifa 07

Diese Anwendungen wurden in der Bakkalaureatsarbeit Analyse des Netzwerkverkehrs von Computerspielen nähre erläutert.

8. Skripte und Programme

Es wurden mehrere Skripte verwendet, um die Tests durchführen zu können. Ein shell–Skript führte die nötigen Nistnet-Befehle aus und ein mgen-Skript markierte den Anfang– und Endpunkt der Szenarien.

8.1. shell–Skripte

Für jedes in Kapitel 5.4 angeführte Szenario gibt es ein shell- Skript. Die folgenden zwei Skripte sind nur Beispiele, um den Aufbau der Skripte zu erklären (es ändert sich immer nur das rot Eingerahmte).
Ablauf:

- In der 4. Zeile wird eine ssh-Verbindung zu PC2 aufgebaut.
- Das shell-Skript “script.sh“ wird ausgeführt (Das shell-Skript startet auf PC2 das mgen-Skript “notraffic.mgen“).
- Anschließend werden die nistnet-Parameter gesetzt nach den 90 Sekunden ist das Testszenario vorüber.

DELAY-Skript

```
echo Verbindungsaufbau zum PC mit der IP 192.168.1.2
echo mgen starten
echo Bitte Warten
ssh root@192.168.1.2 '/root/Desktop/Bakk/script.sh'&
echo Verbindung aufgebaut skript gestartet und verbindung wieder abgebrochen
echo Nistnet wird gestartet
sleep 30
cnistnet -a 192.168.1.5 192.168.0.5 --delay 100
cnistnet -u
echo nistnet gestartet
sleep 60
echo nistnet wird beendet
cnistnet -r 192.168.1.5 192.168.0.5 --delay 100
cnistnet -d
echo nistnet beendet
```

DROP-Skript

```
echo Verbindungsaufbau zum PC mit der IP
 192.168.1.2
echo mgen starten
echo Bitte Warten
ssh root@192.168.1.2 '/root/Desktop/Bakk/script.sh'&
echo Verbindung aufgebaut skript gestartet und verbindung wieder abgebrochen
echo Nistnet wird gestartet
sleep 30
cnistnet -a 192.168.1.5 192.168.0.5 --drop 10
cnistnet -u
echo nistnet gestartet
sleep 60
nistnet wird beendet
cnistnet -r 192.168.1.5 192.168.0.5 --drop 10
cnistnet -d
echo nistnet beendet
```

script.sh

Das Skript startet das mgen-Programm mit dem notraffic-Skript (siehe folgenden Absatz)

```
echo notraffic.mgen
/root/Desktop/Bakk/mgen input /root/Desktop/Bakk/notraffic.mgen &
echo Verbindung wird beendet
exit
echo Verbindung beendet
```

8.2. mgen – Skripte

- **Notraffic.mgen**

```
0.0 ON 2 UDP SRC 5001 DST 192.168.0.2/5000 PERIODIC [1 30]
120.0 ON 1 UDP SRC 5001 DST 192.168.0.2/5000 PERIODIC [1 30]
1 OFF 2
121 OFF 1
```

Kein Hintergrundverkehr, es werden nur am Anfang und Ende die Markierungspakete geschickt.

8.3. Java Programme

Mit einem selbstentwickelten Java-Programm wurden die Daten ausgewertet. Damit dieses Java- Programm richtig arbeiten konnte, musste man mit Hilfe eines WinDump und trpr Aufrufs die Trace - files bzw. Plotfiles erstellen. Diese Files dienen dem Java-Programm als Eingabefiles. Das Java-Programm ermittelt die Paketanzahl und Paketgröße.

9. Testergebnisse

Die Testergebnisse ergaben sich aus den vier getesteten Szenarien und der darauf ausgeführten mgen-Skripten. Es zeigten sich unterschiedlichste Reaktionen bei den Spielen. Diese Situationen wurden während des Spiels protokolliert und ausgewertet.
Im Voraus war sichergestellt worden, dass bei allen getesteten Spielen ausschließlich UDP-Pakete verwendet und versendet wurden.
In den nächsten Kapiteln werden die Testergebnisse der einzelnen Spiele beschrieben und erklärt.

9.1. Age of Empire

9.1.1. Delay

Im Diagramm aoe_delay1 erkennt man, dass die Paketgröße beinahe konstant 13 Bytes beträgt. Sie variiert lediglich um 0,75 Byte. Durch das Betrachten des Diagramms aoe_delay1 erkennt man keinen Trend erkennen, dass bei größeren Verzögerungen (delay) die Paketgröße nur geringfügig zu- bzw. abnimmt. Deshalb ist die Schwankung der durchschnittlichen Paketgröße vernachlässigbar.

Im Diagramm aoe_delay2 hingegen ist ein eindeutiger Trend ablesbar. Bei zunehmender Verzögerung nimmt die Paketanzahl stark ab. Nachdem die Paketzahl anfangs steigt fällt sie gegen Ende stark ab: unter 300 Pakete pro Testfall.

Wenn Delay (Verzögerung) auftritt versucht das Spiel das Netzwerk zu entlasten indem es die Bandbreite verringert. Diese Reaktion auf Verzögerung die von den Essembled Studios integriert wurde versucht vorzeitig den Verkehr zu verringern, sodass kein Stau entsteht. Dasselbe Verhalten besitzt auch Age of Mytologie von welchen der Netcode übernommen wurde.

9.1.2. Drop

Die Summe der Pakete ändert sich kaum bis zu einer Verlustrate von 60 % (siehe Diagramm aoe_drop2). Bei weiter zunehmender Verlustrate erhöht sich die Anzahl der Pakete stark und steigt auf bis zu 340 Pakete pro Testlauf.
Die Paketgröße hingegen ändert sich über den gesamten Testverlauf nicht(siehe Diagramm aoe_drop1).

Die Reaktion auf diese Stauung ist auch hier wieder ähnlich wie jene von Age of Mytology: Das Spiel versucht durch massives Senden von Paketen den Verlust auszugleichen. Die Folge ist die Vergrößerung der Verstopfung.

9.1.3. Throughput

Betrachtet man die Bandbreiten Sending-rate und Throughput ist eine eindeutige Staustrategie des Spiels interpretierbar. Während dem gesamten Testverlauf nimmt die Bandbreite der Sending-rate kontinuierlich zu und die Bandbreite Throughput kontinuierlich ab. Diese Strategie, die hier verwendet wird, ist nicht zielführend. Der Stau bleibt bestehen. Die benötige Bandbreite verringert sich leicht während des gesamten Testdurchlaufs (aoe_delay3).

9.2. Anno 1701

9.2.1. Delay

Bis zu einer Verzögerung von 2 Millisekunden schwindet die Paketgröße. Bei weiterer Erhöhung des „delay“ nimmt sie stark zu. Insgesamt variiert sie bis zu 40 Byte (siehe Diagramm a_delay1).
Aus dem Diagramm a_delay2 kann man erkennen, dass die Anzahl der Pakete über den gesamten Testverlauf stark schwankt. Bei einem Testlauf von 120 Sekunden fällt eine Schwankung von 50 Paketen nicht ins Gewicht und hat somit keine Auswirkung auf die Bandbreite des Spiels.

9.2.2. Drop

Dem Testergebnis zufolge ist ein klarer Trend sichtbar. Abgesehen von einem Ausreißer von 70% nimmt die Paketgröße nimmt bei zunehmender Verlustrate ab (siehe Diagramm a_drop1).

Die Paketanzahl variiert. Trotzdem ist ein steigender Trend erkennbar. Bis zu einer Verlustrate von 60 % sinkt die Paketanzahl um 60 Pakete pro Testfall. Bei zunehmender Verlustrate nimmt die Paketanzahl stark zu. Ziemlich genau um ein Drittel: 80 Pakete (siehe Diagramm a_drop2).

9.2.3. Throughput

Die Bandbreite (sending-rate) nimmt bis zu einer Verlustrate von 30 % ab. Ab diesem Wert bleibt sie relativ konstant. Die Verlustrate nimmt dagegen ständig zu. Hierbei ist ein leichtes Schwanken beobachtbar. Der Durchsatz (engl. „throughput“) nimmt kontinuierlich ab (siehe Diagramm a_drop3).
Anno 1701 weist bei Paketverlust eine andere Reaktion auf wie Age of Empires III. Was die Veränderung der sending-rate betrifft, unterscheiden sich die Spiele deutlich. Bei Age of Empire III nimmt sie zu, bei Anno 1701 bleibt sie, ab einer Verlustrate von 30%, konstant. Im Diagramm a_delay3 befindet sich die Kurve der Bandbreite zwischen 20 – 30 kByte.

9.3. Dawn of War

9.3.1. Delay

Aus unserem Testdiagramm (dawn of war delay1 und –delay2) kann man ablesen, dass sich die Paketgröße geringfügig ändert je größer die Verzögerung ist (Zunahme bis zu 10 Byte). Die Schwankung der Paketzahl pro Testfall ist bei Dawn of War vernachlässigbar: Bei einer Testdauer von 90 Sekunden beträgt sie maximal 20 Pakete.

9.3.2. Drop

Beträgt die Verlustrate unter 50% steigt die Paketgröße um 14 Byte. Ist sie größer nimmt die Paketgröße ihr ursprüngliches Ausmaß wieder an (siehe Diagramm dow_drop1). Da die Kurve nahezu symmetrisch ist ähnelt sie einer Glockenkurve.

Die Paketanzahl nimmt sehr schnell ab und erreicht bei einer Verlustrate von 20 % ihr Minimum. Anschließend steigt sie ungleichmäßig wieder an und erreicht schließlich beinahe die ursprüngliche Größe. Bemerkenswert ist, dass die durchschnittliche Paketgröße und die Paketanzahl gegensätzlich verlaufen. Größere Unregelmäßigkeiten weist jene Kurve auf, welche den Verlauf der Paketanzahl indiziert.

9.3.3. Throughput

Bis zu einer Verlustrate von 60 % sinkt der Durchsatz leicht, anschließend ist ein deutlich stärkeres Abfallen zu beobachten (Siehe Diagramm dow_drop3). Bei diesem Testlauf büßt der Empfänger die Hälfte der ursprünglichen Bandbreite ein. Die Sending-rate steigt leicht an und fällt anschließend wieder auf die Ausgangsgröße.
Relic Entertaiment ist es gelungen bei der Entwiclung von Dawn of War den Netcode so zu gestalten, dass dieser auch mit großen Verlustraten zu Recht kommt.
Die Sending-rate wird kaum verändert, der Durchsatz aber sinkt stark ab. Die Anwendung verkraftet den Datenverlust gut und ein Weiterspielen des Spieles ist möglich. Während des Testlaufs steigt die Sending-rate um 10 kByte und nähret sich, gegen Ende, dem ursprünglich Wert wieder an (siehe Diagramm delay_3).

9.4. Quake

9.4.1. Delay

Im Diagramm q4_delay1 ist ein eindeutiger Trend erkennbar. Die Paketgröße nimmt über den gesamten Testverlauf um das Zweieinhalbfache zu, von 100 Byte auf 250 Byte. Die Zunahme der Paketgröße ist proportional zur Zunahme des Delays.

Aus dem Diagramm (q4_delay2) ist ersichtlich, dass während des Testfalls bei einer Verzögerung von 1000 ms eine besondere Situation aufgetreten sein muss. Es wurden über 100 Pakete mehr versendet, daraus kann man schließen, dass es sich um einen Ausreißer handelt. Eine bestimmte Spielsituation oder hardwarespeziefische Gründe könnten die Ursachen für diesen Ausreißer sein. Über den gesamten Testverlauf blieb die Anzahl der Pakete konstant.

9.4.2. Drop

Während des Testlaufes schwanken Paketzahl und durchschnittliche Paketgröße sehr stark. Es lässt sich kein Trend erkennen, die Paketgröße bewegt sich ständig zwischen 100 Byte und 84 Byte. Die Paketanzahl liegt im Intervall 1650-1700 Pakete pro Testfall (siehe Diagramm q4_drop2).

9.4.3. Throughput

Durch die steigende Verlustrate sinkt der Durchsatz konstant. Er erreicht schließlich die Hälfte der ursprünglichen Bandbreite. Bei Verlust steigt die Sending-rate während des gesamten Testlaufs nicht, sondern schwankt geringfügig (siehe Diagramm q4_drop3). Bei Verzögerung hingegen steigt die Sending-rate bei dem gesamten Testlauf konstant an(siehe Diagramm q4_delay_3).
Wie man im folgenden Bild erkennen kann, ist die Quake-Engine ständig weiterentwickelt worden. Die Entwicklungsgeschichte der ursprünglich von Doom3 übernommenen Quake4-Engine ist im folgenden Diagramm dargestellt.

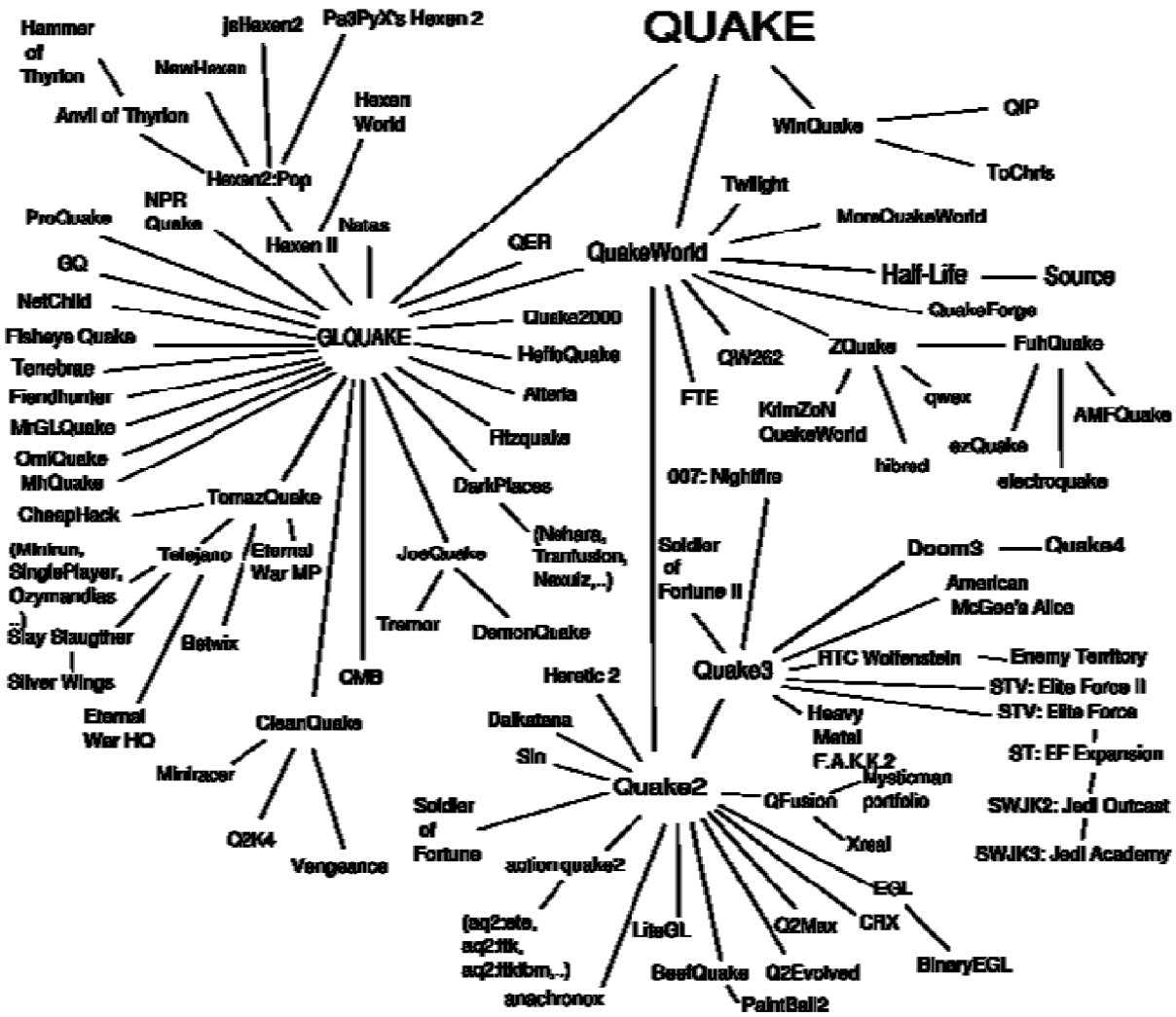


Abbildung 11: Entwicklungsbaum der Quakeengine

9.5. Battlefield 2

9.5.1. Delay

Bis zu einem Delay von 1000 ms bleibt die durchschnittliche Paketgröße konstant bei 444 Bytes. Anschließend fällt die Paketgröße rapide ab und normalisiert sich erst wieder am Ende des Tests. Die Paketgröße bewegt sich in einem Intervall von 40 Byte (siehe Diagramm b2_delay1).

Die Anzahl der Pakete verhält sich in etwa gleich wie die Paketgröße. Bis zu einem delay von 1000 ms bleibt die Paketanzahl konstant. Bei zunehmender Verzögerung nimmt die Paketanzahl stetig ab.

Ab einem delay 2000 ms wurde die Verbindung zwischen Server und Client unterbrochen und der letzte Testfall mit 3000 ms konnte nicht mehr durchgeführt werden.

9.5.2. Drop

Das Intervall der Paketgröße beträgt 360-420 Byte. Bei einer Verlustrate von 30-70 % sind die Pakete am kleinsten: 360 Byte. Bei einer Verlustrate von 90 % springt die Paketgröße auf 420 Byte an (siehe Diagramm b2_drop1). Über den gesamten Testzeitraum kann man erkennen, dass die Paketgröße bei zunehmender Verlustrate abnimmt. Der letzte Testfall konnte nicht durchgeführt werden, da bei einer Verlustrate von 90 % ein Verbindungsabbruch stattfand.

Die Paketanzahl schwankt während des gesamten Testdurchlaufs sehr stark und pendelt sich gegen Ende auf 2468 Pakete pro Tests ein. Die Paketanzahl ändert sich maximal um 18 Pakete (siehe Diagramm b2_drop2) und ist angesichts einer Gesamtheit von 2468 Paketen vernachlässigbar.

9.5.3. Throughput

Die Sending-Rate fällt während des gesamten Testdurchlaufs leicht ab. Bei einer Verlustrate von 80 % steigt sie wieder auf die ursprüngliche Bandbreite von 980 KByte pro Testfall. Der Durchsatz hingegen fällt gleichmäßig um ein Drittel ab (siehe Diagramm b2_drop3), bei Verzögerung verhalten sich die Bandbreiten ähnlich: Sie fallen etwa um ein Viertel ab. Dieses Verhalten trägt, wie die Spiele Dawn of War und Quake 4, nichts zur Staubewältigung bei. Der Netcode wurde von dem Vorgängerspiel dem Battlefield übernommen, welches nach laut Internetrecherche ähnliches Verhalten aufweist.

9.6. Fifa 07

9.6.1. Delay

Bei zunehmender Verzögerung (100-300 ms) sinkt die Paketgröße, steigt anschließend stark an und pendelt sich schließlich bei der doppelten Ausgangswert ein. Die durchschnittliche Paketgröße bewegt sich zwischen 55 und 130 Byte. Die letzten beiden Testfälle (2000 bzw. 3000 ms Verzögerung) führen beinahe zu denselben Testergebnissen. Daraus lässt sich schließen, dass die Staureaktion ausgereizt ist und nicht stärker darauf reagiert werden kann (Siehe Diagramm fifa_delay1).

Beim Diagramm fifa_delay2 ist ein erhebliches Abfallen der Paketanzahl zu beobachten. Je größer die Verzögerung desto geringer ist die Paketanzahl. Bei geringen Verzögerungen sinkt die Paketanzahl schnell, bei größeren Verzögerungen sind hingegen kaum Reaktionen zu beobachten. Die Paketzahl ist bei den Testfällen um bis zu 900 Pakete gesunken.

9.6.2. Drop

Bei Verlust von Paketen ist eine andere Reaktion zu beobachten als bei Verzögerung. Die Paketgröße sowie auch die Paketzahl nehmen bei zunehmender Verlustrate zu, sie verdoppeln sich. Bei einer Verlustrate von 90% unterbrach die Verbindung, weshalb das letze Ergebnis dieser Versuchsreihe aussteht (Siehe Diagramm fifa_drop1 und fifa_drop2).
Obwohl die Anzahl der Pakete stark abnimmt und die Paketgröße zunimmt steigt die benötigte Bandbreite um etwa 10 kbyte.

9.6.3. Throughput

Man kann erkennen, dass der eingehende Datenstrom zunimmt, bis er sich schließlich verfünffacht hat. Der ausgehende Datenstrom bleibt relativ konstant und schwankt nur leicht. Diese Reaktion auf Verlust ist eher Stauhemmend als Stauauflösend. Ab einer Verlustrate ab 50-60 % und Verzögerung von 300ms nimmt die Sending-Rate sehr stark zu, sodass eine Stausituation unvermeidbar ist.

10. Zusammenfassungen

Auffallend ist, dass die Kurven von Sending-rate und Throughput, bis auf minimalste Abweichungen, bei allen fünf Spielen nahezu ident verlaufen (siehe Diagramm delay3).

10.1. Zusammenfassung Age of Empires 3

Age of Empires III benötigt eine sehr kleine Bandbreite, die Paketgröße bleibt bei beiden Versuchsreihen konstant. Das Spiel besitzt eine durchdachte Flussstrategie, sprich, bei Staugefahr wird die Bandbreite verringert.

10.2. Zusammenfassung Anno 1701

Anno 1701 besitzt eine andere Strategie wie Age of Empires III. Bei einer Verzögerung bis zu 2 Sekunden nimmt die Paketgröße ab, anschließend zu. Der Umgang mit großen Verzögerungen gelingt nicht so gut wie bei Age of Empires III, die Staustrategie ist um einiges ausgereifter. Die Sending rate nimmt bei zunehmenden Verlust bis zu 30 % ab und bleibt anschließend konstant. Diese Staubewältigungsstrategie scheint sehr effektiv zu sein, da diese Anwendung den Verkehr reduziert und somit dazu beiträgt, dass sich der Datenstau auflöst.

10.3. Zusammenfassung Dawn of War

Dawn of War reagiert auf Stau. Mit großen Verzögerungen (delay und drop) kann das Spiel gut umgehen. Die Sending-rate wird bei sehr großen Verlustraten nicht verringert und verschlechtert die im Netzwerk auftretende Stausituation nicht. Der Netcode ist auch bei großer Verzögerung stabil und verändert die Bandbreite minimal. Bei den Testfällen ergab sich, dass Dawn of War das Spielgeschehen am längsten aufrecht erhalten konnte.

Trotz intensivster Internetrecherche fanden wir keine negative Kritik. Es uns jedoch nicht möglich in Erfahrung zu bringen ob dieser Netcode auch bei anderen Spielen verwendet wird.

10.4. Zusammenfassung Quake 4

Auf Verzögerung reagiert Quake 4 eindeutig, die Paketgröße vergrößert sich gleichmäßig und die Anzahl der Pakete bleibt konstant. Bei Verlust lässt Quake 4 sowohl die Anzahl der Pakete als auch ihre Größe schwanken. Das Spiel behält die Sending-Rate auch im Staufall bei und wird nicht aktiv um Stauungen zu bewältigen. Der Datenfluss verhält sich ähnlich wie bei Dawn of War, nur dass Quake den 4 fache Kapazität aufweist.

10.5. Zusammenfassung Battlefield 2

Bei Verzögerung sinken Paketgröße sowie Paketanzahl. Das Spiel scheint kaum auf die Verzögerung zu reagieren. Bis zu einer Verzögerung von 1000 ms ist überhaupt keine Reaktion zu beobachten. Anschließen sieht es so aus als ob die Kommunikation einfach solange reduziert wird bis die Verbindung unterbricht. Da die Pakete in Verzögerung eintreffen, verlangsamt sich das Spiel, sodass der notwendige Datenaustausch abnimmt. Deswegen reduziert das Spiel Größe und Anzahl der Pakete. Bei Verlust hingegen ist eine intensivere Reaktion zu beobachten. Die Paketanzahl sowie –Größe ändern sich. Die Paketgröße sinkt anfangs und steigt anschließend, die Paketanzahl schwankt nur geringfügig. Battlefield 2 weist den größten Datenfluss und die schlechteste Staureaktion auf. Die letzten Testfälle (Verzögerung von 3000 ms und Verlustrate von 90%) konnten wegen einem Verbindungsabbruch nicht vollständig durchgeführt werden.

10.6. Zusammenfassung Fifa 07

Fifa 2007 zeigt zwei deutliche Reaktionen auf Stau. Bei Verzögerung wird die Paketanzahl verringert und die Paketgröße vergrößert. Bei Verlust wird beides (Anzahl und Größe) erhöht und somit steigt die Datenverkehr stark an, wobei der Datenfluss nach dem Router nahezu konstant bleibt. Fifa 2007 ist das einzige Spiel, bei dem der Durchsatz (throughput) nicht abgenommen hat. Diese Strategie scheint laut den Ergebnissen eine effiziente Strategie zu sein, allerdings muss man bedenken, dass bei einem realen Datenstau ein solches Verhalten von mehreren Anwendungen die Datenleitung völlig überlasten würde.

Fifa 2007 und Battlefield 2 stammen vom denselben Entwickler von EA (Electronic Arts), weisen aber eine komplett unterschiedliche Stauverhalten auf.
In einem Netzwerkspiel, wo mehrere Clients interagieren, könnte ein solches Stauverhalten, sehr schnell zu einer totalen Verstopfung des Netzwerks führen.

Schlusswort

In dieser Arbeit wurde gezeigt wie Computerspiele auf Verlust und Verzögerung reagieren. Dazu wurde ein Versuchsaufbau entwickelt, der den Datenverkehr zwischen zwei Spielen aufzeichnen und unterschiedliche Stausituationen erzeugen kann. Durch unterschiedliche Testfälle sollte eine Reaktion der Spiele erzwungen werden. Durch die normierten Testläufe erhielten wir unter denselben Rahmenbedingungen Ergebnisse, die einheitlich interpretiert wurden.

Als Testpool wurden inhaltlich unterschiedliche Spiele ausgewählt, welche die drei wichtigsten Spielgattungen exemplarisch repräsentieren:

Strategie:

- Age of Empires III
- Anno 1701
- Dawn of War

Shooter:

- Battlefield 2
- Quake 4

Sport:

- Fifa 07

Keines der Spiele reagierte gleich. Die unterschiedlichen Strategien sind auch nicht auf die Art des Spieles zurückzuführen. Die Entwickler haben bei den Spielen einen entweder schon existierenden oder eine ganz neu entwickelnden Netcode verwendet. Egal ob Strategie- Shooter- oder Sportmultiplayerspiel hat jedes seine eigene Art auf Verlust und auf Verzögerung u reagieren. Es gibt keine einheitliche Staubewältigungsstrategie oder Flusskontrolle für Multiplayerspiele. In einem Netzwerk in dem jeder Client dieselbe Bandbreite erhält kann ein gut programmierter Netcode ein flüssiges Spielen länger aufrecht halten als ein Spiel ohne Staubewältigungsstrategie oder Flusskontrolle.

Quellenverzeichnis

http://www.gnuplot.info
http://lartc.org/lartc.html
http://mgen.pf.itd.nrl.navy.mil/mgen.html
http://www.tcdump.org
http://pf.idt.nrl.navy.mil/protools/trpr.html
http://www.wikipedia.org
http://www.wincap.org
http://wirshark.org

Abbildungsverzeichnis

Danksagung

Für die Betreuung und Unterstützung während der Arbeit danken wir Dr.-Ing Michael Welzl. Für die Bereitstellung der Teststrecke danken wir Sven Hessler und dem Institut für Informatik in Innsbruck.