„Digitale Archivierung und netzwerk-basierte Distribution von Audio-Inhalten“

von

Florian Krause, M.A. aus Berlin

von der Fakultät 1 – Geisteswissenschaften

der Technischen Universität Berlin zur Erlangung des akademischen Grades

Doktor der Philosophie

-Dr. Phil-

Genehmigte Dissertation

Promotionsausschuss:

Vorsitzender: Prof. Dr. W. Hendricks

1. Gutachter: Prof. Dr. W. Sendlmeier

2. Gutachter: Priv. Doz. Dr. Gerrit Kalkbrenner

Tag der wissenschaftlichen Aussprache: 20. Januar 2006

Berlin 2006

D 83

Danksagung

Besonderer Dank gebührt folgenden Personen für ihre Hilfe bei der Erstellung dieser Arbeit:

- Jimmy Warwas, Dominic Schuster und Stefan Zapf, Warptec GmbH für die Hilfe bei den PHP und Perl Skripten
- Stefan Fellenberg für die Bereitstellung der benötigten Hardware
- Ralf Richter von der Spreeradio GmbH für die Beschreibung der eingesetzten Speichersysteme bei diesem Sender

Stilistische Konventionen

Um diese Arbeit besser lesbar zu machen, werden besondere Abschnitte und Ausdrücke durch unterschiedliche Schriftarten dargestellt:

- Quellcode: Auszüge aus dem Quellcode werden `in Courier New, Schriftgröße 12` dargestellt.

- Programmnamen: Namen von Programmen und Skripten werden *kursiv* dargestellt.

Weiterhin wurde, um die Länge der Fußnoten am Seitenende etwas einzugrenzen, auf die komplette Angabe der Quelle (falls diese in Buchform vorliegt) im Text verzichtet und stattdessen ein Kürzel in der Form [WAL2000] verwendet. Die vollständigen Angaben dazu sind unter Abschnitt 7.6 nachzulesen.

Abstract

Diese Arbeit beschäftigt sich mit der Archivierung von Audiodaten in digitaler Form sowie der Distribution dieser Daten über digitale Netzwerke. Eines der Hauptziele ist dabei die Konzipierung, Analyse und Entwicklung eines Softwaresystems, welches es dem Benutzer ermöglicht, Musikstücke, Wortbeiträge, Nachrichten und andere Inhalte sehr kostengünstig über ein IP-basiertes Netzwerk an andere Teilnehmer zu übertragen.
Dies soll durch den kombinierten Einsatz verschiedener Übertragungstechniken weitestgehend ohne Streaming realisiert werden.

This thesis deals with the digital storage of audio data and its distribution over networks. One of the main goals is the design, analysis and development of a system which provides an inexpensive possibility to deploy music, recorded speech, news and other content over an IP-based network. This goal should be achieved without using data streaming by means of various transmission methods.

1. Hintergrund der Arbeit

Diese Dissertation ist als Fortführung meiner Magisterarbeit mit dem Titel "Konzeption und Realisierung eines internet-basierten Radio-Automationssystems[1]“ zu betrachten. Diese entstand zwischen 2002 und 2003 unter der Betreuung von Prof. Dr. Manfred Krause. Von ihm stammt auch die Anregung für das erstellte Distributionssystem dieser Arbeit.

Mein Interesse am Thema Internet-Audio begann, als ich während meines Studiums für etwa zwei Jahre als Netzwerkadministrator für ein Internetradio tätig war und dabei stark von der Möglichkeit als Privatperson einen Radiosender betreiben zu können, fasziniert wurde.

Ziel der damaligen Magisterarbeit war, eine frei verfügbare Software zum Gestalten eines Onlinesenders zu entwickeln und die damit verbundenen Aufgabenstellungen, Technologien und Probleme zu beleuchten. Die daraus gewonnen Ergebnisse dienen als Grundlage für die vorliegende Arbeit.

[1] http://www.tektribe.de/magisterarbeit.pdf

2. Einleitung

Bei der im Rahmen der Magisterarbeit geschehenen Portierung eines Radio-Automationssystems in eine internet-basierte Plattform konnten die meisten Probleme in Bezug auf die Bedienbarkeit und Zuverlässigkeit eines solchen Systems zufrieden stellend gelöst werden. Als problematischer erwiesen sich dagegen die Einschränkungen des Internets beim Streaming von Audio- und Videodaten. Vereinfacht ausgedrückt handelt es sich dabei um drei Faktoren, die den Einsatz eines solchen Systems im großen Stil verhindern.

- Schlechte Skalierbarkeit: Die im Internet hauptsächlich nutzbare Punkt-zu-Punkt Übertragung führt dazu, dass die Anforderungen an die eingesetzte Hardware linear mit der Anzahl der Hörer steigen.
- Hohe Betriebskosten: Der erste Punkt führt wiederum dazu, dass ein System mit vielen Nutzern nicht mehr wirtschaftlich rentabel umzusetzen ist.
- Ungenügende Redundanz: Größere Störung an der Quelle des Datenstroms führen unweigerlich zu Ausfällen von Bild und/oder Ton bei allen verbundenen Nutzern.

Die vorliegende Arbeit behandelt die Frage, ob es durch den kombinierten Einsatz von Multicast, Pull-Technologien, Peer-to-peer Übertragungen und Distributed Computing gelingt, die Skalierbarkeit und Redundanz von Audio- und Videoanwendungen im Internet nachhaltig zu verbessern. Das in diesem Rahmen entwickelte Softwaresystem DIRA soll dabei als Proof-of-Concept für die im Folgenden aufgestellten Thesen dienen:

These 1. Individuelle Anforderungen für den Radiobetrieb sind mit Internettechniken realisierbar.

These 2. Die Übereinstimmung von Meta-, Audio- und Videodaten auf örtlich verschiedenen Speichersystemen kann durch vorhandene Netzwerkprotokolle sichergestellt werden.

These 3. Die Distribution eines Radioprogramms über das Internet ist auch bei weitgehendem Verzicht auf Streaming realisierbar.

These 4. Durch die kombinierte Nutzung von Multicast, Pull-Technologien, Peer-to-peer Übertragungen und Distributed Computing lassen sich die Probleme bei der Übertragung von Audio und/oder Bildmaterial im Internet umgehen.

These 5. Für den Betrieb eines solchen Systems sind Computer aus dem Low-Cost Segment ausreichend.

These 6. Durch Nutzung von Pull-Technologien kann die zu übertragende Datenmenge sowie die Last auf dem Server stark reduziert werden.

These 7. Durch den Einsatz von Peer-To-Peer Technologien werden die Datenmenge und die Last auf dem Server nochmals weiter reduziert.

These 8. Durch den Einsatz von verteilten Rechnerstrukturen ist eine Erhöhung der Betriebsstabilität erzielbar.

These 9. Durch die kontrollierte Vervielfältigung von Musiktiteln ist die rechtliche Situation anders als etwa bei Tauschbörsen einwandfrei geklärt.

2.1. Digitale Archivierung

Um die mit dieser Arbeit verbundenen Problemstellungen besser beleuchten zu können, betrachten wir zunächst die vorhandenen digitalen Archivierungsmethoden. Dabei beschränken wir uns auf „State of the art“ Technologien, denn diese sind weit verbreitet und haben ihre Funktionalität im alltäglichen Einsatz bewiesen.

Wir werden zwischen Speicherung auf proprietären Datenträgern (z.B. CD, DAT) und Formaten ohne spezielles Speichermedium (z.B. MP3, PCM) unterscheiden.

2.2. Digitale Datenträger

Betrachtet man die geschichtliche Entwicklung der Speicherkapazität von digitalen Datenträgern, so stellt man fest, dass die Musikindustrie technologisch lange Zeit wesentlich fortschrittlicher als die Computerindustrie war. So wurde das erste digitale Trägermedium zur Speicherung von Audiodaten, die Compact Disc (oder einfacher CD) bereits im Jahr 1980 von den Firmen Sony und Philips standardisiert[2]. Sie bot zur damaligen Zeit eine enorm hohe Klangqualität. Pro CD konnte man eine Datenmenge von 650 MByte speichern - eine gigantische Menge, wenn man bedenkt, dass Festplatten dieser Zeit über eine Kapazität von gerade einmal 10 MB verfügten[3].

Im Jahr 1983 waren die ersten CD-Player marktreif und haben seitdem analoge Medien wie die Langspielplatte oder die Musikkassette beinahe komplett vom Markt verdrängt. Der Erfolg dieses Mediums war überwältigend: Im Jahr der Einführung konnten 800.000 Datenträger verkauft werden, 1990 waren es bereits 288 Millionen und im Jahr 2000 mehr als 1 Milliarde Einheiten. Da die Audio-CD auch heute noch das am häufigsten eingesetzte Speichermedium für Musik ist, werden wir im nächsten Abschnitt auf deren Funktionsweise eingehen.

---

[2] http://www.oneoffcd.com/info/historycd.cfm

[3] [SIA2001], S. 172 - 179

2.2.1. Audio Compact Disc

Bei der Audio-CD handelt es sich um eine 12 cm große und 1,2 mm dicke Scheibe aus Polycarbonat, welche als Träger für die Daten dient. Der Lesevorgang findet spiralförmig auf einer Spur von innen nach außen statt, wobei die Rotationsgeschwindigkeit nach außen hin gedrosselt wird. Die Lineargeschwindigkeit beträgt so konstant 1,2 m/s.

Entlang der etwa 0,5 µm breiten Spur befinden sich seriell angeordnete Vertiefungen (Pits), welche ebenfalls 0,5 µm breit sind und dabei eine Tiefe von 128 nm und eine minimale Länge von 0,83 µm besitzen. Der Abstand zwischen den Spuren beträgt 1,6 µm.

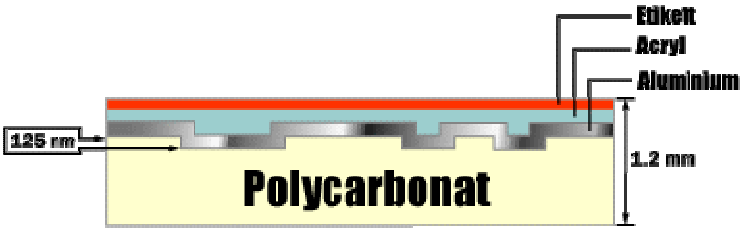

Abbildung 2.2.1a - CD im Querschnitt (nicht maßstabsgetreu)

In Abbildung 2.2.1a wird deutlich wie eine CD produziert wird. Der noch transparente Polycarbonat-Rohling wird mit Hilfe eines Master-Rohlings geprägt, wodurch die mikroskopisch kleinen Vertiefungen an dessen Oberseite entstehen. Diese wird mit einer optisch reflektierenden Aluminiumschicht überzogen, darauf folgt eine Acrylschicht (oder auch Nitrolack) gegen Kratzer. Der Begriff Pits bezieht sich auf eine Vertiefung in der Oberfläche einer CD, wenn man diese von der bedruckten Seite aus betrachtet. Für den Laser, der von unten auf die CD zugreift, handelt es sich um Erhöhungen[4]. Das Gegenteil eines Pits (also der Orginal-Zustand des Trägers) wird als Land bezeichnet.

[4] [SCW1995], S. 16 - 18

Um die auf einer CD gespeicherten Daten auszulesen, wird ein Laser mit einer Wellenlänge von 780 nm verwendet. Die Abstrahlleistung beträgt zwischen 0.5 und 1.0mW. Leistungsregelung und Ausgleich von spannungs- oder thermisch bedingten Schwankungen erfolgen über eine direkt in den Halbleiter integrierte Monitorfotodiode mit angeschlossener Vergleichsschaltung. Das emittierte Licht ist monochrom und kohärent. Diese Eigenschaften der Lichtquelle, gleiche Wellenlänge und gleiche Phasenbeziehung, werden im CD-System ausgenutzt. Dabei wird die Intensität des Lichtes durch Interferenzen mit dem reflektierten Signal moduliert. Auf diese Weise kann das gespeicherte Signal rekonstruiert werden.

Das Abspielsystem fokussiert den Durchmesser des Laser-Lichtpunktes an der Aluminiumschicht auf 1,7 µm, was etwa der dreifachen Pitbreite und der doppelten minimalen Pitlänge entspricht. Dies wird durch den Brechungsindex des Trägerkunststoffes Polycarbonat begünstigt, welcher den an der Unterseite immerhin noch 0,8 mm großen Durchmesser des Lichtpunktes auf die gewünschte Größe fokussiert (siehe Abb. 2.2.1b).

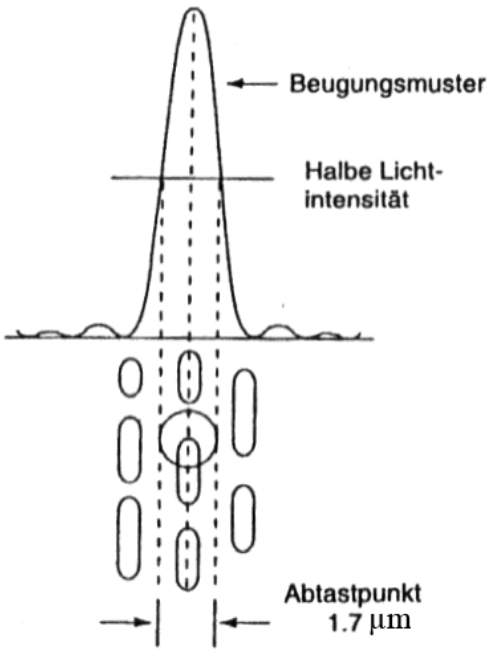

Abbildung 2.2.1b - Fokussierung des Laserstrahls an der Aluminiumschicht

Die Justage von Fokus und Spur erfordert bei einer derart kleinen Spurbreite ein äußerst präzise arbeitendes Abtastsystem. Dieses Pick-Up genannte System versucht Ungenauigkeiten, die etwa durch Vibrationen oder Erschütterungen des CD-Spielers aber auch durch unsauber produzierte CDs entstehen können, zu erkennen und abzufangen. Dazu werden zwei unabhängige Regelschaltkreise zur horizontalen und vertikalen Korrektur eingesetzt.

Bei optimaler Spurführung trifft der Laser nun entweder zu 100 % auf ein Land oder teilweise auf ein Pit und wird dabei von der Aluminiumoberfläche reflektiert. Je nach Pitlänge werden somit zwischen 30 % und 50 % des Lichts vom Pit reflektiert, der Rest vom umgebenden Land, welches im Strahlengang weiter entfernt liegt. Bei einer Pittiefe von konstant 128 nm sind also ein pit- und ein landreflektierter Strahl um etwa 230 nm "ortsverschoben", was etwa λ/2 der resultierenden Wellenlänge des Lasers im Polycarbonat entspricht. Dieses hat einen Brechungsindex von 1.55, somit ergibt sich eine Wellenlänge von 503 nm im Kunststoff.

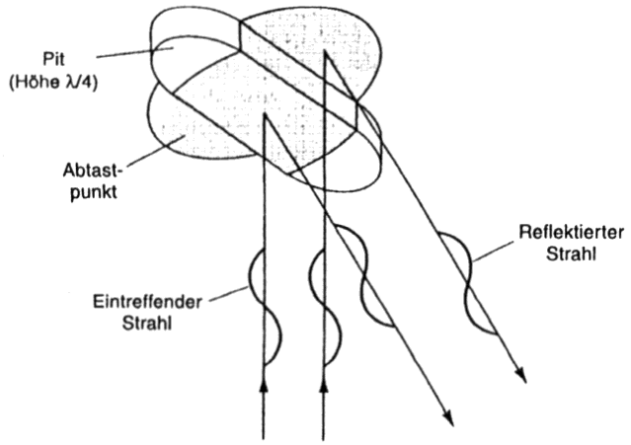

Abbildung 2.2.1c - Phasenverschiebung durch Pit-Reflektion

Führt man diese Strahlen wieder zusammen, so interferieren sie destruktiv und löschen sich weitgehend gegenseitig aus. Da der Strahl nur vereinzelt zu gleichen Teilen auf Pit und Land trifft, kommt es selten zu einer vollständigen Auslöschung

sondern meistens zu einer Intensitätsmodulation des reflektierten Lichtbündels. Dieses wird über einen Photosensor in elektrische Signale umgewandelt und den Schaltkreisen des CD-Players zugeführt[5].

Hier beginnt der digitale Teil der Signalverarbeitung. Das Aufnahmeverfahren der CD ist nicht so einfach, dass ein Pit eine logische 1 und ein Land eine logische 0 bezeichnet. Vielmehr wird ein Non-Return-To-Zero Code eingesetzt, bei dem das Kanal-Bit mit dem Wert 1 durch den Übergang von Pit zu Land oder Land zu Pit dargestellt wird. Findet kein Übergang statt, so wird dies als Kanal-Bit mit dem Wert 0 interpretiert. Pits und Lands müssen eine Länge von mindestens drei und höchstens elf Kanal-Bits haben, da sonst die sichere Erkennung der Übergänge nicht gewährleistet werden kann.

Aus diesem Grund wird bei der CD das Eight-To-Fourteen Verfahren eingesetzt, welches aus den 8 Datenbit 14 Kanal-Bits macht. Die Dekodierung erfolgt über in der Hardware des CD-Spielers eingebaute EFM-Tabellen. Ein Auszug daraus:

Dezimal	Binar	EFM
00	0000 0000	01001000100000
01	0000 0001	10000100000000
02	0000 0010	10010001000000
...	...	...

Tabelle 2.2.1a - EFM Kodierung

Wie man sieht, wird durch die EFM-Kodierung eine Häufung von 0 vermieden, außerdem treten nach jeder 1 mindestens zwei 0 auf.

Vor der weiteren Verarbeitung der Audiosignale werden Maßnahmen zur Fehlerverdeckung und Fehlerkorrektur getroffen. Hierzu fügt man den Quelldaten weitere Datenbits hinzu, verschachtelt die Daten untereinander und verschlüsselt die gesamte Datenstruktur nach einem bestimmten Code. Als Fehlerkorrekturcode wird

[5] [COJ1997], S. 208 - 209

ein doppelter Blockcode verwendet. Das Verfahren zur Blockerzeugung ist nach den Erfindern Reed-Solomon benannt (Cross Interleaved Reed-Solomon, kurz CIRC).

Dabei werden 12 16-bit Werte (entsprechend 6 Stereo-Abtastwerten) zu einem Block von 24 8-bit Werten umsortiert. Diesen 24 Symbolen werden durch einen Paritätsgenerator 4 weitere Symbole (Q-Parität) hinzugefügt, um sie im Anschluss durch eine zeitliche Codespreizung zu verschachteln. Die Spreizung soll bewirken, dass sich die Pits und Lands eines Datenwertes über eine größere Fläche der CD verteilen. Die 28 verschachtelten Symbole werden einem zweiten Paritätsgenerator (C1-Codierer) zugeführt, welcher das Signal um vier P-Symbole erweitert. Somit ergibt sich folgender schematischer Aufbau:

<table>
<tr><td colspan="4">1 Frame, entspr. 256 bit</td></tr>
<tr><td>Informationsbits
12 Symbole zu je 8 bit = 96 bit</td><td>CIRC
Q-Parität
4 Symbole</td><td>Informationsbits
12 Symbole zu je 8 bit = 96 bit</td><td>CIRC
P-Parität
4 Symbole</td></tr>
<tr><td colspan="4">0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31</td></tr>
</table>

Tabelle 2.2.1b - Audiodaten-Frame

Aus den P und Q Symbolen ergibt sich eine Prüfsumme für die Datenzeilen bzw. Spalten.

Neben den Audio- und Fehlerkorrekturdaten sind weiterhin noch Zusatzinformationen wie Spieldauer eines Titels, Titelnummer, Synchronisationssteuerung sowie alphanumerische Informationen zur Anzeige auf dem Display gespeichert. Dazu wird jedem Frame ein weiteres Control & Display genanntes Datenbyte angehängt. Für etwa die Anzeige des Titelnamens ist dieses Byte jedoch nicht ausreichend, daher werden je 98 zu einem so genannten Subcode zusammengefasst[6,7,8].

[6] http://www.tu-chemnitz.de/informatik/RA/kompendium/vortraege_96/CDROM/cdrom0.html

[7] http://www.muenster.de/~asshoff/physik/cd/cdplayer.htm

[8] http://www.disc4you.de/kompendien/cd/aufbau/logi.htm

Für die abschließende digital-analog Wandlung wird bei aktuellen CD-Playern in der Regel ein 1-bit Verfahren eingesetzt. Dieses Verfahren kann an folgendem Beispiel verdeutlicht werden: Durch sehr schnelles Betätigen eines Lichtschalters kann man in einem Raum verschiedene Helligkeitsstufen erzeugen. Ähnlich verhält es sich bei der 1-bit Wandlung: Die bei der CD benötigte Taktrate von 2,9 GHz (44,1 kHz * 2^{16}) ist allerdings erst mit Signalprozessoren moderner Bauart kostengünstig realisierbar. Der Vorteil dieses Verfahrens gegenüber der Multi-Bit Wandlung ist, dass man dadurch Quantisierungsfehler in der Amplitudendarstellung vermeidet.

Abschließend sei erwähnt, dass die CD in ihrer ursprünglichen Form keinesfalls als Archivierungsmedium geschaffen wurde. Um eine CD zu produzieren benötigt man Produktionsstätten mit Reinräumen und entsprechend teuren Maschinen. Auch ist die Idee des Masters, welcher zur Vervielfältigung eingesetzt wird, widersprüchlich zu einer Kleinserie oder Privatkopie. Erst mit der Erfindung des CD-Brenners 1990 (ebenfalls durch Philips) änderte sich dies. Mit diesen Geräten wurde es dem Privatanwender möglich, selber Daten auf einem so genannten CD-Rohling zu sichern. Dazu wird eine organische Substanz (meist Cyanin oder Phtalocyanin) durch einen Laser beim Beschreiben so verändert, dass dadurch die Pits und Lands simuliert werden können. Aus diesem Grund können auch normale CD-Spieler gebrannte CDs abspielen[9]. Für die professionelle Archivierung eignen sich diese Datenträger auf Grund ihrer relativ hohen Fehleranfälligkeit und beschränkten Lebensdauer jedoch nicht[10].

Mittlerweile werden in Deutschland mehr (leere) CD-Rohlinge als bespielte CDs verkauft[11]. Dieser Umstand veranlasste die Musikindustrie dazu, CDs zu verkaufen, welche nicht mehr dem CD-Standard entsprechen. Diese lassen sich daher nicht mehr in einem Computer abspielen, sondern nur noch in einem HiFi-Gerät. Dies liegt daran, dass die Computer-Laufwerke neben Audio- auch noch Daten-CDs

[9] http://stud3.tuwien.ac.at/~e0125556/privat/studium/cdbrenner.html

[10] [GRA2000], S. 114ff

[11] http://www.welt.de/data/2003/08/15/153186.html

lesen können. Im Daten-Bereich einer solchen CD befinden sich ungültige Angaben, so dass ein Computer die CD nicht erkennt – die primitiveren Laufwerke in Audio-CD Spielern dagegen ignorieren diese Fehler meistens.[12,13]

2.2.2. Digital Audio Tape

Neben der Audio-CD ist das Digital Audio Tape, kurz DAT, das einzige digitale Endverbraucher-Medium, bei dem keine Datenkompression zum Einsatz kommt (siehe Abs. 2.3.2)[14]. Das System wurde 1987 von Sony vorgestellt und sollte die digitale Nachfolge der Musikkassette antreten.

Das DAT-Format gab dem Benutzer erstmals die Möglichkeit, Daten digital auf ein Band aufzuzeichnen - mit einstellbarer Sample- (32, 44,1 und 48 kHz) und Bitrate (13 und 16 Bit). Dazu bediente man sich der Helical-Scan Technik, die schon beim Videorekorder erfolgreich zum Einsatz gekommen war. Bei dieser Technologie werden die Daten nicht linear (also längs der Bandrichtung) aufgezeichnet, sondern schräg (siehe Abb.2.2.2a). Dies wurde notwendig, da die hohe Datendichte bei linearer Aufzeichnung eine extrem hohe Bandlaufgeschwindigkeit erfordert hätte.

---

[12] http://www.heise.de/ct/cd-register/default.shtml

[13] [MCR1994], S. 25 - 29

[14] http://atknoll1.informatik.tu-muenchen.de/Zope/tum6/lectures/seminars/ws0203/avformate/bierbaum.pdf

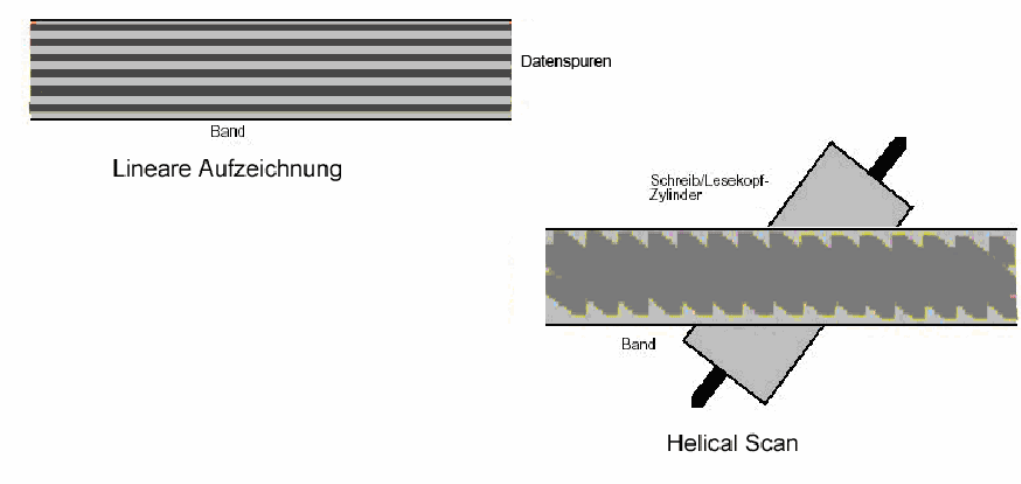

Abbildung 2.2.2a - Lineare Aufzeichnung und Helical Scan

Hier ist zu erkennen, dass das Lesen und Schreiben der Spur durch einen schräg zum Band stehenden Zylinder, welcher je zwei Lese- und Schreibeinheiten enthält, geschieht. Da der Zylinder mit 2000 U/min entgegen der Bandlaufrichtung (8,15 mm/s) rotiert, wird bei jedem halbem Umlauf die nächste Quer-Spur des Bandes gelesen oder geschrieben. Anders als in der vereinfacht dargestellten Grafik, ist die Breite der Spuren sehr klein (13,591 µm), so dass durch diese Technik eine hohe Datendichte erreicht werden kann. Die relative Geschwindigkeit beträgt 3,1 km/s (!), kein existierendes Material dürfte die nötige Beschleunigung im linearen Aufzeichnungsbetrieb aushalten.

Im Detail stellt sich die Spuraufzeichnung folgendermaßen dar:

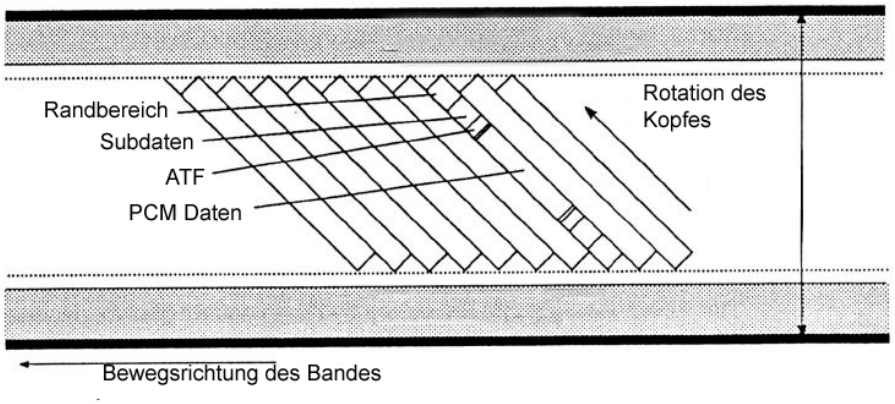

Abbildung 2.2.2b - Aufbau der Schrägspuren

Eine Spur beginnt und endet in einem ungenutzten Randbereich, der den vollen Kontakt des Kopfes mit dem Band sicherstellen soll. Darauf folgt der Bereich für Subcode-Daten (Absolutzeit, Start- oder Stop-Indizes usw.) und der Automatik Track Following Bereich, in dem Informationen für die Synchronisation des Bandlaufes und der Trommel abgelegt sind.

Die Speicherung der Daten erfolgt wie bei der CD durch einen PCM Code in der Mitte der Schrägspur. Jede Spur besteht aus 196 Blöcken, wovon 128 Blöcke die Audiodaten speichern. Da ein Block aus 288 Bit besteht (wovon 256 Bit nutzbar sind), können 1024 Stereosamples pro Spur aufgezeichnet werden. Als Kodierung kommt eine 8 nach 10 Modulation zum Einsatz, bei der maximal 3 gleiche Zeichen nacheinander auftreten können.

Dezimal	Binär	8 nach 10
00	0000 0000	0101010101
01	0000 0001	0101010111
02	0000 0010	0101011101
...	...	...

Tabelle 2.2.2a - 8 nach 10 Kodierung

Für die Fehlerkorrektur kommt wie bei der Audio-CD ebenfalls der Reed-Solomon Algorithmus zum Einsatz.

Obwohl das DAT-System die Aufnahme von Musik in sehr guter Qualität erlaubt, konnten nie große Stückzahlen der Geräte abgesetzt werden. Lediglich im Studiobereich hat sich das Format als Standard durchgesetzt. Der Grund dafür liegt wohl hauptsächlich darin, dass die Musikindustrie erkannte, das verlustfreie Kopien eine große Gefahr für ihren Markt darstellen und darauf drängte, die Geräte mit einem Kopierschutz auszustatten. Letztendlich gelangten nur solche auf den Markt, die mit dem Serial Copy Management System (SCMS) ausgestattet waren. Dieses System schränkte die Funktionen der DAT-Rekorder stark ein - so war etwa nur eine digitale Kopie eines Bandes möglich, anfangs war sogar die Aufnahme mit 44,1 kHz untersagt. Es ist unverständlich warum bei der Einführung der beschreibbaren CD, welche ja eine noch wesentlich einfachere und schnellere Vervielfältigung von Audio-CDs erlaubt, durch die Verantwortlichen der Musikindustrie keine entsprechenden Gegenmaßnahmen eingeleitet wurden[15,16].

2.2.3. Minidisc

Die Minidisc wurde etwa zur gleichen Zeit (1992) wie die digitale Audiokassette (DCC) auf den Markt gebracht. Wie die CD basiert das von Sony entwickelte Verfahren auf eine mit konstanter Winkelgeschwindigkeit rotierenden Scheibe, welche aber zum besseren Schutz vor Beschädigungen in einem Plastikgehäuse (Caddy) untergebracht ist.

Die Scheibe hat einen Durchmesser von 64 mm und ist 1,2 mm dick - auf Grund der geringeren Größe ergibt sich eine Speicherkapazität von 140 MB. Es gibt drei verschiedene Arten von Minidiscs:

---

[15] http://www.kgw.tu-berlin.de/~cbradter/labor2001/Band.pdf

[16] [COJ1997], S. 50 - 53

- Bespielte Kauf-Minidiscs: Diese entsprechen technisch weitestgehend der Audio-CD. Auf einer Aluminiumschicht sind Erhebungen aufgebracht, die durch das Laser-Pick-Up System eingelesen werden. Dabei wird eine Eight-to-Fourteen Modulierung sowie ein Reed-Solomon Fehlerkorrekturalgorithmus eingesetzt. Auch die Wellenlänge des Lasers, die Spurbreite sowie der Spurabstand sind identisch. Bespielte Minidics sind jedoch kaum noch im Handel erhältlich und die Verkaufszahlen daher stark rückläufig[17].
- Wiederbeschreibbare Minidiscs: Anders als beim CD-R Format kommt bei dieser Variante ein magneto-optisches Verfahren zum Einsatz, auf das wir gleich eingehen werden.
- Hybrid Minidiscs: Diese enthalten sowohl einen vorgefertigten als auch einen beschreibbaren Bereich. Diese Variante wurde kaum hergestellt und ist nicht mehr im Handel erhältlich.

Das Verfahren zum Beschreiben und Löschen von Minidiscs wurde von der magneto-optischen CD (CD-MO) abgeleitet. Dabei befindet sich auf einer beschreibbaren Disc eine vorgefertigte Spur, welche spiralförmig von innen nach außen verläuft. Die Ränder dieser Spur sind gewellt, so dass durch die Wobble-Modulation in einem 22,5 kHz Trägersignal Adreßinformationen gespeichert werden können. Diese erleichtern dem Gerät das Einstellen der korrekten Winkelgeschwindigkeit und die Positionierung des Schreib-/Lesekopfes.

Die Aufzeichnung der Informationen erfolgt durch die Magnetisierung des vom Laser erhitzten Materials. Hierbei befinden sich Lasereinheit und Magnetkopf gegenüber, die Disc liegt dazwischen (siehe Abb. 2.2.3a).

[17] http://www.ifpi.de/jb/2003/32-37.pdf

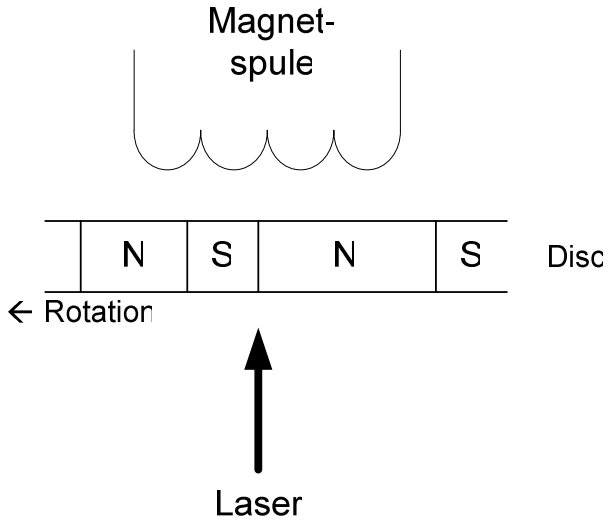

Abbildung 2.2.3a - Schematische Darstellung Minidisc MO-System

Der Laser mit einer Leistung von etwa 4,5 mW erhitzt die Disc punktuell auf über 180° C. Dadurch wird die magnetische Polarisierung (falls die Minidisc bereits bespielt war) aufgelöst und sogleich durch das Magnetfeld der Spule neu ausgerichtet. Verlässt der Laser die Stelle, so kühlt das Material wieder ab und die Polarisierung "erstarrt" im letzten Zustand. Durch diese Ausrichtung des magnetischen Feldes können 0 und 1 repräsentiert werden. Durch sehr schnelles Umschalten des Feldes kann die minimale Länge einer Informationseinheit auf 0,6 µm gesenkt werden.

Zum Lesen der Daten wird nur die Lasereinheit benötigt. Die verwendete Energie ist wesentlich geringer (< 1mW), so dass die Magnetisierung beim Auslesen nicht verändert wird. Auf Grund des Kerr-Effekts hat die magnetische Polarisierung der Minidisc einen Einfluss auf die Polarisierung des reflektierten Laserlichts. Auf diese Weise kann das magnetische Feld wieder in binäre Daten umgewandelt werden[18,19].

Da eine Minidisc nur etwa 1/5 der Speicherkapazität einer CD besitzt, beide Systeme aber die gleiche Aufnahmedauer pro Medium bieten, ist es erforderlich,

[18] http://home.t-online.de/home/richard.buechter/theorie.html

[19] http://www.physik.uni-bielefeld.de/~hempel/diplom/node50.html

dass auf der Minidisc die Daten in einer komprimierten Form gespeichert werden. Das hierbei verwendete proprietäre Verfahren der Firma Sony nennt sich ATRAC (Adaptive Transform Acoustic Coding). Es soll aber nicht näher behandelt werden, da es in der Funktionsweise der MPEG-Audio Layer-2 Kodierung stark ähnelt (siehe Abs. 2.3.4). Auf Grund der geringen Rechenleistung der ersten Minidisc-Geräte war das zugrunde liegende psychoakustische Modell stark vereinfacht. Daher rührt auch noch der schlechte Ruf bei Audiophilen. Mittlerweile ist ATRAC in der Version 4.5 auf dem Markt, welche eine sehr gute Klangqualität bietet - bei Kompatibilität zu älteren Geräten[20]!

Die Minidisc konnte anfangs große Erfolge verbuchen und ließ die DCC-Systeme (siehe nächster Abs.) schnell hinter sich. Mit dem Aufkommen von immer billigeren CD-Brennern hat das Format allerdings an Bedeutung verloren, was wohl an der wesentlich größeren Verbreitung von CD-Playern liegt[21,22]. Ein anderer Faktor dürfte sein, dass Minidisc-Systeme keine digitalen Kopien erlauben (ausgenommen einige sehr teure, professionelle Systeme). Überspielt man einen Titel von Minidisc zu Minidisc, so kann der ATRAC-Kodierer mit dem datenreduzierten Signal nicht sehr gut umgehen und bereits ab der 5. Kopie ist eine Verschlechterung des Klangbildes wahrnehmbar[23]. Zwar findet man dieses Verhalten auch etwa bei MP3-Kodierern, da Dateien dieses Formats aber digital kopierbar sind, stellt dies kein Problem dar. Eventuell war dieser Effekt bewusst vom Hersteller Sony in Kauf genommen worden, um die Verbreitung von Titeln durch Raubkopien einzuschränken[24].

---

[20] http://www.minidisc.org/faq_sec_4.html

[21] http://www.oto-online.com/june00/foc_mini.html

[22] http://www.ifpi.de/jb/2003/32-37.pdf

[23] http://www.minidisc.org/faq_sec_4.html#_q23

[24] [COJ1997], S. 174 - 175

2.2.4. Weitere Datenträger

Neben der Audio-CD, der Minidisc und dem Digitale Audio Tape existieren noch eine Reihe von anderen digitalen Datenträgern, auf die hier aber nicht im Detail eingegangen werden soll.

- Digital Compact Cassette (DCC): Dabei handelt es sich um eine Weiterentwicklung des DAT Verfahrens mit feststehenden Köpfen (S-DAT). Um die Bandlaufgeschwindigkeit niedrig zu halten, setzten die Hersteller Philips und Mathushita auf das Precision Adaptive Subband Coding Verfahren, welches kompatibel zu MPEG-1 Layer 1 ist. Die Produktion von DCC-Systemen wurde bereits 4 Jahre nach der Veröffentlichung 1992 wieder eingestellt, da sich die konkurrierende Minidisc durchgesetzt hatte[25,26].
- ADAT: Bei diesem von der Firma Alesis entwickelten Verfahren kommen S-VHS Kassetten, welche über einen rotierenden Lese- / Schreibkopf gelesen und beschrieben werden, zum Einsatz. Es wird vor allem im Studiobereich eingesetzt, da man 8 Spuren gleichzeitig aufnehmen und wiedergeben kann. Die Samplingrate beträgt 44,1 oder 48 kHz, die Wortbreite 16 Bit.
- TA88: Dieses von der Firma Tascam entwickelte Verfahren ähnelt ADAT stark, hier werden allerdings 8 mm Videobänder benutzt. Es bietet ebenfalls 8 Spuren.
- DASH: Bei diesen Geräten handelt es sich um die digitale Umsetzung der im Studiobereich sehr weit verbreiteten analogen Bandmaschinen mit festem Lese- / Schreibkopf (z.B. Studer). Sie können bei 1/2 Zoll Bandbreite (in diesem Fall die physische Abmessung des Bandes) bis zu 48 Spuren verarbeiten.

[25] http://come.to/dcc-faq

[26] [COJ1997], S. 59

Es ist anzumerken, dass viele Fernseh- und Radiostationen mittlerweile dazu übergegangen sind, Titel nur noch auf Computer-Datenträgern zu archivieren [27]. Wir werden solche Systeme in folgenden Abschnitten weiter behandeln.

2.3. Technik zur digitalen Archivierung von Audiodaten

Durch die enorme Leistungssteigerung bei Computern sowie den drastischen Preisverfall bei den Datenträgern wurde etwa Mitte der neunziger Jahre die Speicherung von Audiodaten auf Festplatte oder ähnlichen Medien für die Audio-Industrie interessant. Durch den Einsatz von datenreduzierenden Verfahren wurde es in den folgenden Jahren sogar möglich, auf handelsüblichen PCs einige tausend Musiktitel zu speichern. Auch im professionellen Bereich (z.B. im Radio) wurde die Produktions-, Distributions- und Sendetechnik nach und nach auf eine ausschließlich digitale Abwicklung ohne externe Datenträger umgestellt (siehe Abs. 2.4). Dies hat folgende Vor- und Nachteile:

Vorteile:

- Beliebig viele verlustfreie, digitale Kopien
- Verlustfreie Übertragung
- Verlustfreie Speicherbarkeit
- Hohe Speicher- und Lesegeschwindigkeit
- Schnellerer Zugriff auf gesuchte Titel
- Unabhängigkeit von einem Datenträger
- Wesentlich preisgünstiger als etwa digitale Bandmaschinen

[27] [WEH1989], S. 60 - 63

Nachteile:

- Transfer zwischen Rechnern unter Umständen schwierig (mittlerweile sind diese Schnittstellenprobleme weitgehend gelöst)
- Copyright-Probleme
- Ausfallsicherung muss durch redundante Hardware gewährleistet werden

Wir werden nun im Einzelnen die verschiedenen Verfahren zur digitalen Speicherung von Audiodaten auf Computern beleuchten.

2.3.1. Unkomprimierte digitale Speicherung

Hierbei handelt es sich um die einfachste Art der Speicherung von Audiodaten auf Computern. Zunächst ist es bei einem analogen Eingangssignal erforderlich, dieses in ein digitales Format umzuwandeln. Dies erfolgt in der Regel durch die Bearbeitungsschritte Abtastung, Quantisierung und Kodierung (siehe Abb.2.3.1a).

Die Klangqualität ist im Wesentlichen abhängig von der Abtast- oder Samplingrate sowie der Wortbreite der Quantisierungswerte. Bei etwa der CD benutzt man 44100 Abtastwerte pro Sekunde, welche jeweils durch einen 16-bit Wert (entsprechend einem Bereich von 0 - 65536) dargestellt werden. Im professionellen Bereich wird häufig mit höherer Samplingrate (48 kHz oder 96 kHz) und 24 oder gar 32 Bit Wortbreite gearbeitet.

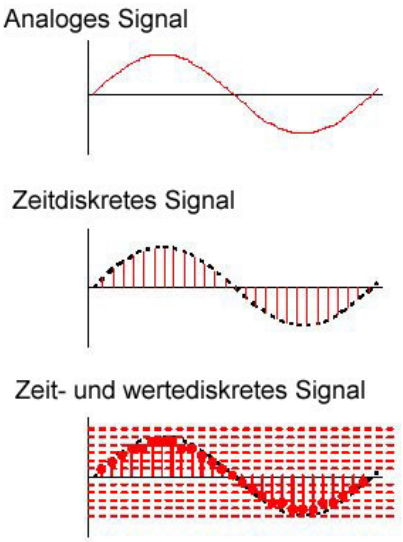

Abbildung 2.3.1a - Analog-Digital Wandlung

Die Anzahl der für die Speicherung benutzten Bits legt fest, wie viele unterschiedliche Werte zur Repräsentation des Analogsignals verwendet werden können. Je weniger Bits zur Verfügung stehen, desto gröber ist die Approximation der Digitalwerte an das Ursprungssignal. Werden diese wieder zurück in ein analoges Signal gewandelt, so ist die Differenz zwischen Original und abgetastetem Signal als Rauschen wahrnehmbar. Diese Erscheinung wird Quantisierungsrauschen bezeichnet. Wie man in Abb. 2.3.1b erkennen kann, ist es umso kleiner, je mehr Quantisierungsstufen vorhanden sind.

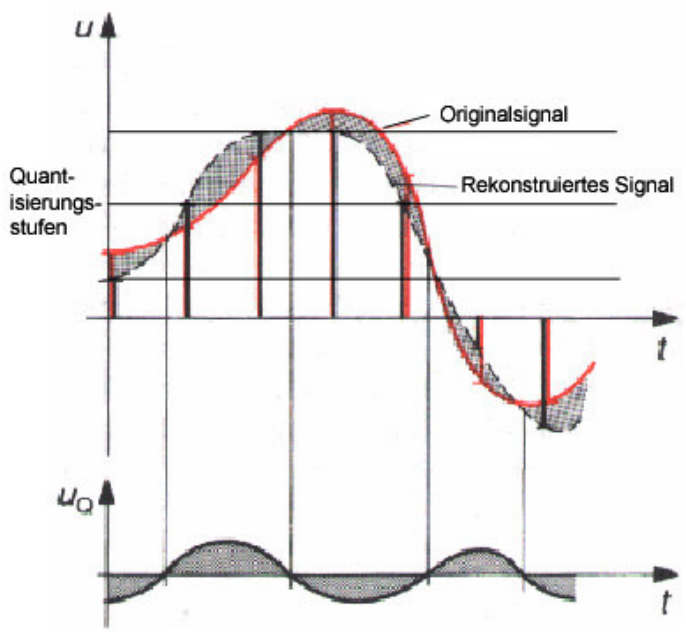

Abbildung 2.3.1b - Quantisierungsrauschen

Bei der Audio-CD mit 16 Bit Wortbreite ergibt sich ein Quantisierungsrauschen, welches bei -96 dB unter Vollaussteuerung liegt. Es dürfte damit in den meisten Fällen nicht mehr wahrnehmbar sein[28].

$$f_{\max} = \frac{f_{Abtast}}{2}$$

Formel 2.3.1a - Nyquist Theorem

Analog zur Anzahl der Bits ergeben sich durch die Samplingrate Einschränkungen für das aufgenommene Signal. Nach dem Nyquist-Theorem gilt, dass die höchste in einem Signal vorkommende Frequenz maximal die Hälfte der Abtastfrequenz betragen darf. Um dies sicherzustellen wird das Eingangssignal durch einen Tiefpassfilter mit $f_{\max}$ als Grenzfrequenz geleitet.

[28] [COJ1997], S. 227 - 228

Das meistgenutzte Verfahren zur Speicherung von Audiosignalen nennt sich Pulse Code Modulation (PCM). Die Quantisierung kann dabei linear (lineare PCM) oder logarithmisch erfolgen (μ-Law oder A-Law). Letztere werden hauptsächlich im Bereich der Telefonkommunikation eingesetzt. Die Anwendung einer logarithmischen Skala führt dazu, dass leise Signale genauer aufgelöst werden als laute Signale[29].

Will man Audiodaten in CD-Qualität auf einer Festplatte speichern, so benötigt man pro Audiokanal und Sekunde eine Bandbreite von 16 Bit * 44100 = 705,6 KBit oder 88,2 KByte. Ein Stereo-Titel mit einer Spieldauer von 5 Minuten würde daher rund 53 MByte belegen.

Für aktuelle Rechner mit oft über 100 GByte Festplattenkapazität stellt diese Datenmenge kein Problem mehr dar. PCM wird daher meist dort eingesetzt, wo es auf möglichst originalgetreue Wiedergabe und samplegenauen Schnitt ankommt. Die gängigen professionellen Schnitt- und Kompositionssysteme arbeiten mit unkomprimierten Audiodaten[30].

2.3.2. Datenkompressionsverfahren

Bereits in den Anfängen der Computertechnik wurde nach Verfahren gesucht, welche die anfallenden Datenmengen verkleinern sollen, sei es um sie besser speichern oder schneller übertragen zu können. Am häufigsten werden dabei die Verfahren von Huffmann oder Lempel-Ziv eingesetzt. Dabei wird mittels statistischer Methoden versucht, Wiederholungen im Originalcode zu erkennen und durch kürzere Zeichen zu ersetzen, was zu einer Erhöhung der Informationsdichte führt. Aus diesem Grund spricht man hierbei von Entropie-Verfahren. Bei Huffmann erhält zum Beispiel das am häufigsten auftretende Zeichen den kürzesten Code. Charakteristisch ist für beide Verfahren, dass die komprimierten Daten exakt

[29] [SCW1995], S. 86 - 87

[30] http://www.tvhandbook.com/support/pdf_files/audio/Chapter8_1.pdf

wieder hergestellt werden können, daher auch die Bezeichnung verlustlose Kompression.

Diese Verfahren eignen sich für Audio- und Videomaterial nur begrenzt, da hier meist eine statistische Gleichverteilung der Datenwerte zu beobachten ist. So konnte eine 57,6 Megabyte große unkomprimierte PCM-Datei mittels des RAR-Packers[31] auf 44,3 Megabyte komprimiert werden (23 % Kompression). Das RAR-Format verwendet dabei einen (unbenannten) Multimedia-Algorithmus, der effizienter als der beim ZIP-Format[32] verwendete Huffman-Algorithmus arbeitet. Dort wurde nur eine Dateigröße von 55,3 Megabyte (4 % Kompression) erzielt[33,34].

2.3.3. Verlustbehaftete Kompression

Die von RAR erzielte Dateigröße ist zwar sehr beachtlich, sie lässt sich aber noch erheblich reduzieren. Dazu bedient man sich der verlustbehafteten Kompression. Bei dieser werden die Unzulänglichkeiten des menschlichen Gehörs ausgenutzt und sämtliche Daten, die nicht wahrnehmbar sind, entfernt. Der Forschungszweig, der sich mit der akustischen Wahrnehmung des Menschen beschäftigt, heißt Psychoakustik. Dort wurde mit Hilfe vieler Hörtests bestimmt, welche Teile eines Signals hörbar sind und welche nicht. Aus diesem Grund bezeichnet man diese Art der Kodierung als Quellkodierung. Durch diese Verfahren, auf die wir im weiteren Verlauf noch näher eingehen werden, kann ein Kompressionsfaktor von bis zu 1:12 erreicht werden. Das populärste Format dürfte das am Fraunhofer Institut in Erlangen[35] von Prof. Brandenburg entwickelte MP3-Format sein. Es bietet bei einer Bitrate von 128 KBit/s annähernd CD-Qualität, bei 192 KBit/s fällt es selbst Fachleuten sehr schwer, das Original von der komprimierten Version zu

[31] http://www.rarsoft.com

[32] http://www.winzip.com

[33] http://www-user.tu-chemnitz.de/~mfie/compproj/index.htm

[34] http://www.winrar.de/html-ger/info/intrview.htm

[35] http://www.iis.fraunhofer.de/amm/techinf/layer3/

unterscheiden[36]. Bei letzterer Bitrate ist die oben erwähnte Beispieldatei 7,83 Megabyte groß, das entspricht einer Verkleinerung von 87 Prozent.

Es sei erwähnt, dass das MP3-Format circa 10 Jahre alt ist - in der Computerbranche ein sehr langer Zeitraum. Mittlerweile gibt es neue Formate, welche eine noch bessere Komprimierung erzielen. Besonders hervorzuheben ist hier das Ogg Vorbis Format, welches bereits bei 128 KBit/s hervorragende Resultate liefert[37]. Zudem ist es im Unterschied zu den meisten anderen Audio-Formaten frei verfügbar und somit ohne Lizenzgebühren nutzbar.

Im weiteren Verlauf soll am Beispiel des MP3-Formats die Funktionsweise von verlustbehafteten Kompressionsverfahren erläutert werden. Prinzipiell unterscheiden sich die anderen Formate (Ogg Vorbis, Real, Windows Media usw.) lediglich durch optimierte psychoakustische Modelle und verbesserte Programmierung, nicht aber in der Arbeitsweise[38]. Ihnen allen liegen so genannte Hybridverfahren zu Grunde, da bei der Komprimierung sowohl Entropie- als auch Quellverfahren zum Einsatz kommen.

Verlustbehaftete Audio-Kompressionsverfahren machen sich die Unzulänglichkeiten des menschlichen Gehörs zu Nutzen und entfernen nicht hörbare Teile aus dem Originalsignal. Zwei Punkte spielen dabei eine besondere Rolle:

1. Simultane Maskierung: Ein spektraler Signalanteil niedrigen Pegels (maskiertes Signal) kann von einem gleichzeitig stattfindenden starken Signal (Maskierer) unhörbar gemacht werden, falls Maskierer und maskiertes Signal eine ähnliche Frequenz haben. Im Alltag wird man dieses Phänomen etwa wahrnehmen, wenn ein Zug an einem vorbeifährt und so sämtliche Unterhaltung unmöglich macht.

---

[36] [ZOV2000], S.152-155

[37] [HAS2002], S. 94-99

[38] http://www.airwindows.com/encoders/

Da in einem Audiosignal in der Regel mehrere Maskierer vorhanden sind, kann man diese zu einer globalen Maskierungsschwelle zusammenfassen. Alle Signalanteile, die unterhalb dieser Schwelle liegen, sind nicht wahrnehmbar und können somit weggelassen werden. Für den Encoder ist somit nur noch der Bereich von der Maskierungsschwelle bis zum lautesten vorkommenden Signalteil von Interesse.

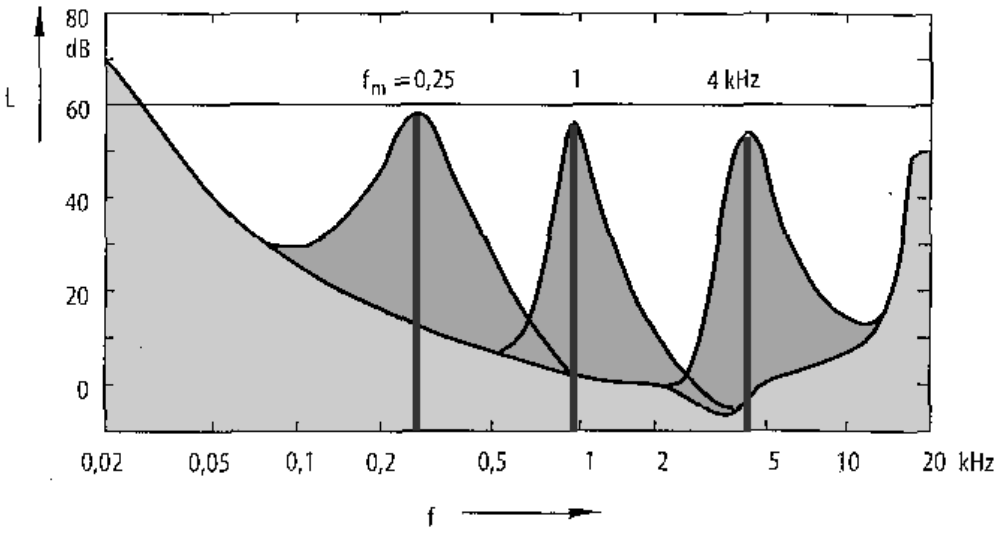

Abbildung 2.3.3a - Simultane Maskierung

In Abbildung 2.3.3a wird dies noch einmal verdeutlicht. Der unterste graue Bereich (hellgrau) wird als Hörschwelle bezeichnet - Signale darunter können vom Ohr nicht wahrgenommen werden. Zusätzlich sind drei 60 dB laute Signale bei 250 Hz, 1000 Hz und 4000 Hz eingezeichnet. Der dunkelgrau eingezeichnete Bereich stellt den durch die drei Signale maskierten und somit unhörbaren Frequenzbereich dar.

2. Temporale Maskierung: Nimmt unser Ohr etwa ein lautes Geräusch wahr, so ist es danach für bis zu 200 ms weniger empfindlich. Kurioserweise tritt die temporale Maskierung auch schon vor dem Geräusch auf, dort allerdings nur in einem wesentlich kürzeren Zeitrahmen. Dies lässt sich auf Grund

unterschiedlicher Verarbeitungswege in der akustischen Physiologie des Menschen erklären.

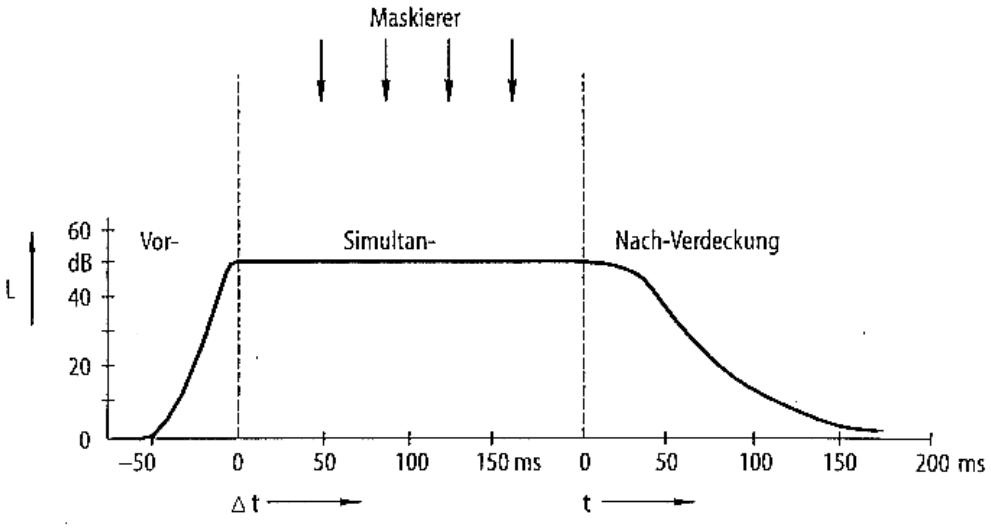

Abbildung 2.3.3b - Temporale und simultane Maskierung

In Abbildung 2.3.3b wird die temporale sowie simultane Maskierung dargestellt. Während des 200 ms langen Signals tritt die oben beschriebene simultane Maskierung auf, bereits 50 ms vor diesem Signal verschlechtert sich die Empfindlichkeit des Gehörs durch die temporale Maskierung, ebenso wie bis etwa 150 ms nach dem Signal[39,40].

2.3.4. Die MPEG-Audio Standards

Die Standards der Moving Pictures Coding Experts Group (MPEG) sind die weltweit am häufigsten benutzten Verfahren zur Kodierung von Bild und Audio.

Bei der Veröffentlichung des MPEG-1 Standards 1992 war absehbar, dass CD-ROM Laufwerke die Diskettenlaufwerke der damaligen Computer ablösen würden.

[39] http://www.fh-wedel.de/~si/seminare/ss02/Ausarbeitung/9.digitalaudio

[40] http://www.ifi.unizh.ch/mml/publications/diplomarbeiten/schrieber.pdf

Mit der CD-ROM stand erstmals ein kostengünstiges, digitales Speichermedium zur Verfügung, auf dem sich ausreichend Daten für digitales Video speichern ließen[41]. Daher war die Zielsetzung von MPEG-1, einen Standard für die Speicherung und Wiedergabe von Bewegtbildern bei einer Auflösung von 352 x 240 Pixel (im NTSC Standard) und einer Datenrate von 1,5 MBit/s (1-fache Laufwerksgeschwindigkeit) zu entwickeln. Er besteht aus vier Teilen:

- Synchronisations- und Multiplexereinheit
- Kompressionsverfahren für Bildmaterial im Progressive-Format
- Kompressionsverfahren für Audio
- Testverfahren zur Sicherstellung der korrekten Datenstrukturen

Der für diese Arbeit interessante Audio-Teil ist wiederum in 3 Unterpunkte gegliedert:

- Layer 1: Beschreibt Verfahren zur verlustbehafteten Audiokompression bei einer Datenrate von 384 KBit/s.
- Layer 2: dito mit einer Datenrate von 192 KBit/s
- Layer 3: Dieses Verfahren, offiziell MPEG-1 Audio-Layer 3 wird allgemein als MP3 bezeichnet. Es bietet bei einer Datenrate von 128 KBit/s CD-ähnliche Klangqualität

Je höher der Audio-Layer, desto komplexer sind die zur Kodierung benötigten Operationen. Man muss anmerken, dass die Geschwindigkeit handelsüblicher Computer zu jener Zeit bei Weitem nicht ausreichte, die enorm umfangreichen Rechenoperationen von MP3 in Echtzeit zu bewältigen. Bei den ersten Versuchen des Autors im Jahr 1994, mit einem AMD 80386 Prozessor mit 40 MHz Taktrate ein Musikstück in MP3 zu verwandeln, dauerte das Kodieren mehrere Stunden, selbst das Abspielen in Echtzeit war nur mit Aussetzern möglich.

[41] [WAJ2001], S. 342-345

Im 2 Jahre später erschienen MPEG-2 Standard wurde der Videobereich um höhere Auflösungen und Bitraten erweitert. Der Audio-Layer erfuhr ebenfalls kleinere Modifikation wie die Definition kleinerer Bitraten und verfeinerter Stereo-Modelle. Bereits 1997 wurde das Nachfolgeverfahren zu MP3 Advanced Audio Coding (AAC) definiert. Erst aktuelle Geräte wie etwa der Apple iPod beherrschen die Wiedergabe dieses gegenüber MP3 wesentlich komplexeren Formates[42,43].

MPEG-2 Audio hat bis heute unverändert eine große Marktbedeutung. Es kommt auf der DVD, beim digitalen Fernsehen und Radio sowie in der Sendeabwicklung von Radio- und Fernsehsendern zum Einsatz[44,45].

2.3.5. MPEG Audio Layer 1 und 2

Die Audio Layer 1 und 2 sind weniger komplex als MP3. Layer 1, welches zum Beispiel auf der Digitalkassette von Philips zum Einsatz kam, ist mittlerweile kaum noch anzutreffen. Layer 2 wird beim digitalen Radio (DAB) eingesetzt und erfreut sich vor allem bei der Archivierung in Radiostationen großer Beliebtheit. Wir werden in Abs. 2.5 beispielhaft ein Archivierungssystem und die dahinter stehenden Backup-Strategie beleuchten.

MPEG Audio Layer 1 und 2 haben gegenüber der MP3-Kodierung folgende Unterschiede:

- Die Frequenzbänder der Eingangsfilterung haben die selbe Bandbreite
- Es findet keine diskrete Kosinustransformation statt
- Keine Huffman-Kodierung der Ausgangswerte
- Nur 512 Band FFT-Analyse (bei Layer 1)

---

[42] http://www.sims.berkeley.edu/courses/is224/s99/GroupG/report1.html

[43] http://www.apple.com/ipod/

[44] [EBU1998], S.51-54

[45] [PEL2000], S. 549 - 553

- Einfacheres psychoakustisches Modell
- Vereinfachte Quantisierungsalgorithmen

Vor allem der Wegfall der Kosinustransformation erspart dem Encoder viel Rechenarbeit und ermöglicht so ein einfacheres Systemdesign[46,47,48].

2.3.6. MP3-Kodierung

Im Folgenden soll das für diese Arbeit sehr relevante MPEG-Audio Layer-3 Verfahren genauern beschrieben werden. Als Ausgangsmaterial für die MP3-Kompression wird in der Regel ein PCM-Audiosignal benutzt. Dabei handelt es sich um ein unkomprimiertes Format mit einer Abtastfrequenz von 44,1 kHz und einer Wortbreite von 16 Bit, wie es etwa bei der Audio-CD zum Einsatz kommt. Damit der Encoder bestimmen kann, welche Teile des Signals hörbar und welche durch die beschriebenen Maskierungseffekte unhörbar sind, ist es erforderlich, dieses von der vorliegenden Zeit- in eine Frequenzdarstellung zu überführen. Dabei wird nicht mehr die Amplitude, sondern die vorkommenden Frequenzen über der Zeit dargestellt. Dazu werden je 1152 Samples eingelesen und mittels einer Filterbank in 32 Frequenzbänder überführt. Die Breite der Frequenzbänder richtet sich dabei nach der Empfindlichkeit des Gehörs, das heißt, in den weniger empfindlichen Bereichen (etwa bei Frequenzen > 16 kHz) ist sie sehr breit und umgekehrt (siehe Abb. 2.3.6a).

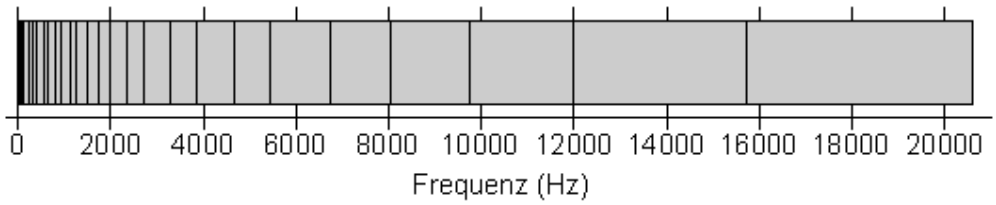

Abbildung 2.3.6a - Frequenzbänder bei der MP3-Kodierung

[46] http://www.gadegast.de/frank/mpegfaq/mpe632.html

[47] http://mail.phys-iasi.ro/Library/Computing/mpeg_layer_3.htm

[48] [POM2001], S. 22 - 24

Um eine noch höhere Frequenzauflösung zu erhalten, werden die Bänder jeweils einer diskreten Kosinus-Transformation unterzogen. So gewinnt man je 18 weitere Bänder. Somit werden die Eingangssamples in 576 verschiedene Frequenzbänder transformiert.

Parallel dazu wird das Eingangssignal einer 1024-band Fast-Fourier-Analyse unterzogen, um für die Bestimmung des psychoakustischen Modells eine noch höhere Frequenzauflösung zu erhalten.

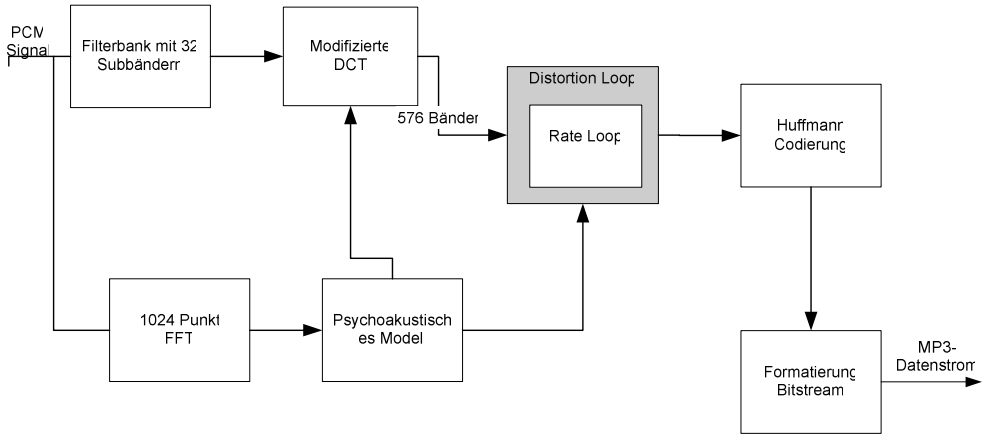

Abbildung 2.3.6b - Blockschaltbild eines MP3-Encoder

Die so gewonnen Daten werden benutzt, um die in Abs. 2.3.3 beschriebenen psychoakustischen Effekte zu bestimmen. Hierbei kann etwa festgestellt werden, dass bestimmte Frequenzbänder komplett unhörbar sind – diese werden dann nicht kodiert.

Eine weitere Kompressionswirkung wird durch die nachfolgende Quantisierung erzielt. Dabei gibt es zwei sich gegenseitig kontrollierende Elemente:

- Rate Loop: Dieser Programmteil betrachtet für jedes Frequenzband die Maskierungsschwelle und den maximalen Pegel. Da alle Signale unterhalb der Maskierungsschwelle nicht hörbar sind, ist nur der Bereich zwischen

dieser und dem Maximalpegel von Interesse. Da dieser, auch Signal-to-Mask genannte, Bereich nicht mit den vollen 16-Bit, sondern auch mit weniger Bit dargestellt werden kann, wird ein kleinerer Quantisierungsfaktor gewählt. Je kleiner dieser Faktor, desto geringer ist die erforderliche Datenrate.

- Distortion Loop: Dieser Programmteil ist der Gegenspieler des Rate Loop. Da sich, wie oben beschrieben, durch einen kleineren Quantisierungsfaktor der Pegel des Quantisierungsrauschens anhebt, versucht dieser Teil, durch einen Skalierungsfaktor die Quantisierung so festzulegen, dass das Quantisierungsrauschen unterhalb der Maskierungsschwelle liegt und somit nicht hörbar wird.

Kodiert man etwa ein Musikstück mit einer sehr niedrigen Bitrate (z.B. 24 KBit/s), so kann man sich das Zusammenwirken der beiden Schleifen etwa so vorstellen:

Die Rate Loop versucht, um sämtliche hörbaren Frequenzbänder in der verfügbaren Bandbreite unterzubringen, sehr große Quantisierungsschritte (z.B. 2 Bit pro Band), da dies zu einer kleinen Ausgangsbitrate führt. Der Distortion Loop dagegen stellt fest, dass das Quantisierungsrauschen in vielen Bändern hörbar ist und versucht dem gegenzusteuern. Beide Schleifen würden nun endlos die Quantisierung (bzw. die Skalierung) vor und zurückstellen, so dass für diesen Fall Abbruchbedingungen geschaffen werden müssen. Ist etwa zu wenig Bandbreite verfügbar, so werden bei MP3 komplette Frequenzbänder entfernt - bei 24 KBit/s alle Signale über 6 kHz. Reicht die Bandbreite dann immer noch nicht aus, so müssen beide Programmteile einen Kompromiss zwischen wahrnehmbaren Verzerrungen und Bandbreite aushandeln. Solch niedrigbitratigen Signale klingen daher auch nicht gut - allerdings immer noch besser, als wenn man einfach die Abtastrate und Quantisierungswortbreite verringern würde[49].

Abschließend werden die Daten noch huffmann-kodiert und in einen MP3-kompatiblen Datenstrom ausgegeben[50]. Da die damit gewonnen Daten sowohl quell-

[49] http://www.iis.fraunhofer.de/amm/techinf/layer3/index.html

[50] [COJ1997], S. 54

als auch entropiekodiert sind, spricht man hierbei auch von einem hybriden Verfahren.

Wenn man bedenkt, dass all diese Berechnungen für jedes der 576 Frequenzbänder und pro Sekunde etwa 38-mal durchgeführt werden müssen (44100 Samples pro Sekunde geteilt durch 1150 Samples pro Block), so kann man verstehen, wie richtungweisend dieses Verfahren bei seiner Entwicklung war. Damalige Rechner benötigten noch ein Vielfaches der zeitlichen Länge eines Titels, um diesen zu kodieren. Eine Anwendung in Echtzeit war nicht möglich. Für aktuelle Geräte stellt das Kodieren von MP3 keine besonders hohe Anforderung mehr dar. So kodiert zum Beispiel ein Athlon 1,3 GHz Prozessor einen 342 Sekunden langen Titel in nur 41 Sekunden - etwa einem Achtel der Spieldauer. Ein noch modernerer Rechner (Athlon 64, 2 GHz Takt) benötigt hierfür nur noch knapp 20 Sekunden.

Da bei der Wiedergabe lediglich die Daten der einzelnen Frequenzbänder synthetisiert und zusammengeführt werden müssen, benötigt man hierfür wesentlich weniger Rechenleistung. Es gibt mittlerweile fertige Chips zur Wiedergabe von MP3-Dateien. Aus diesem Grund ist in letzter Zeit eine Vielzahl von MP3-fähigen Geräten (USB-Speichersticks, MP3 CD-Player usw.) auf den Markt gekommen[51].

2.4. Systemgestaltung für die professionelle Archivierung von Audio

Will man in einem professionellen Umfeld, wie etwa einem Radiosender ein Musik- und Beitragsarchiv anlegen, wird man dafür wesentlich mehr Aufwand betreiben als etwa ein Privatanwender, der seine Schallplattensammlung digitalisiert und auf CD brennt. Ein solches professionelles System muss dabei folgende Kriterien erfüllen:

[51] http://www.atmel.com/dyn/resources/prod_documents/doc4109.pdf

- Korrektheit: Die auf dem System eingesetzte Software muss die beabsichtigten Aufgaben mit der erforderlichen Genauigkeit ausführen (z.B. Einhaltung der Latenzzeiten).
- Robustheit: Das System darf durch äußere Störungen (z.B. Fehlbedienung durch den Anwender) nicht negativ beeinträchtigt werden.
- Ausfallssicherheit: Defekte an der Hardware dürfen nicht zum Ausfall des Systems führen.

Vor allem die ersten beiden Punkte sind durch die hohe Komplexität der eingesetzten Betriebssysteme und Programme nur annähernd erzielbar. Indem man die verwendeten Komponenten auf bestimmte Einsatzgebiete festlegt, verringert man die Wahrscheinlichkeit, dass das komplette System von einem Fehler betroffen ist. Im Bereich der Archivierung unterscheidet man dabei zwischen folgenden Bereichen:

- Play-Out Archiv: In diesem Archiv befinden sich die für das aktuelle Programm benötigten Audiodateien. Die Funktionalität dieses Systems muss unter allen Umständen mit denen im weiteren Verlauf beschriebenen Verfahren sichergestellt werden.
- Datenbank: Hier werden die zu den Titeln erfassten Metadaten gespeichert. Da die hierfür benötigte Datenmenge eher gering ist, kann die Datenbank auf dem Play-Out Archivsystem gehalten werden.
- Langzeit-Archiv: Hier befinden sich die nicht mehr im aktuellen Programm eingeplanten Titel. Der Umfang dieses Archivs hängt stark von der Art des Senders ab. So wird etwa die Sicherung aller Beträge des Deutschlandfunks erheblich mehr Aufwand erfordern, als etwa für einen Hit-Sender nötig ist.

Für den Sendebetrieb ist noch eine Reihe von weiteren Systemen nötig (z.B. Senderechner, Programmplanung, usw.), die aber hier nicht näher erläutert werden sollen.

Für einen Radiosender ist nicht nur das ausfallsichere Archivieren von Titeln wichtig, sondern auch das der entsprechenden Metadaten. Das können neben den offensichtlichen Daten wie Künstler, Name des Titels und Albumtitel auch Werte wie Geschwindigkeit, Genre, Zielgruppe, weiblicher oder männlicher Sänger, Veröffentlichungsjahr, GEMA Nummer, Häufigkeit der Wiederholungen usw. sein. Da die Sender bemüht sind, ein möglichst großes Zielpublikum anzusprechen, ist eine ausgewogene Abstimmung des Musikprogramms essentiell. Daher wird viel Aufwand und Geld in die Programmplanung gesteckt. Die oben erwähnten Metadaten helfen der Programmplanungssoftware (bei Spreeradio kommt zum Beispiel *RDS Selector* zum Einsatz[52]), um ein ausgewogenes und abwechslungsreiches Programm zu gestalten. Um die Datensätze besser indizieren zu können, wird dabei meist auf die Speicherung in einer zentralen Datenbank zurückgegriffen[53].

Im Folgenden sollen Techniken zur Erhöhung der Ausfallsicherheit erörtert und am Beispiel des Radiosenders Spreeradio erklärt werden:

Bei diesem Sender wird die Software *Zenon* der Firma Zenon Media GmbH eingesetzt[54]. Dabei handelt es sich um eine sehr mächtige (und teure) Software zur vollständig digitalen Sendeabwicklung. Sie speichert die Audiodaten im MPEG-Audio Layer 2 Format. Die Metadaten werden in einer SQL-Datenbank festgehalten. Dieses digitale Archiv bildet das Herzstück des Senders - fällt es aus, muss auf Notbetrieb mit CDs oder Band umgestellt werden. Um eine sehr hohe Verfügbarkeit zu erreichen, setzt man, neben dem Einsatz von sehr hochwertiger und teurer Hardware, auf eine Vielfach-Redundanz Strategie:

- RAID-Subsysteme: Hierbei werden mehrere Festplatten zu einem Verbund zusammengeschaltet. Sie geben sich gegenüber dem Computer als ein Laufwerk aus. Im einfachsten Fall wird eine Absicherung durch die Spiegelung zweier Platten erzielt (RAID-1). Meist kommt RAID-5 zum

[52] http://www.rcseurope.de/produkte/home.html

[53] [MCR1994], S. 154 - 166

[54] http://www.zenon-media.com/

Einsatz, bei dem mindestens drei Festplatten benötigt werden. Diese werden alternierend beschrieben, sollte ein Laufwerk ausfallen, können dessen Daten aus den auf den verbleibenden Platten gespeicherten Prüfsummen wiederhergestellt werden. Diese Systeme sind "hot-pluggable", d.h. es ist möglich, sie im laufenden Betrieb auszuwechseln[55]. Häufig wird auch ein zusätzlicher Ersatzdatenträger (Hot Spare) verwendet, welcher im Havariefall eine ausgefallene Festplatte automatisch ersetzt. Im beschriebenen Sender kommen ausschließlich SCSI-Festplatten in einem RAID-5 Verbund mit zusätzlichem Hot-Spare Laufwerk zum Einsatz.

- Redundante Hardware: Ausfallssichere Festplatten bieten keine Sicherheit, wenn der dazugehörige Rechner ausfällt. Die Lüfter in Netzteil und auf dem Prozessor sind dabei am häufigsten betroffen, da sie durch mechanischen Abrieb Alterungserscheinungen ausgesetzt sind. Daher verwendet man Gehäuse mit zwei Netzteilen (ebenfalls hot-pluggable) sowie Mainboards für zwei (oder mehr) Prozessoren. Fällt ein Prozessorlüfter aus, wird der betroffene Prozessor deaktiviert, das System kann aber noch weiter betrieben werden. Eine weitere Form der Redundanz bezieht sich auch auf das Vorhandensein von zusätzlichen Rechnern, die durch manuelles Umschalten die Funktion eines Havariegerätes übernehmen können. So existieren bei Spreeradio mehrere Standby-Rechner, die durch das Einspielen der aktuellen Sicherungen in kurzer Zeit zu etwa einem Dateiserver oder Ausspieler konfiguriert werden können.

- Cluster: Selbst die relativ geringe Wahrscheinlich eines Hardwarefehlers an Mainboard, Prozessor, Speicher usw. ist relevant. Aufgrund von Alterungserscheinungen wird jedes ununterbrochen laufendes System früher oder später Fehler zeigen. Um selbst den Komplettausfall des Servers kompensieren zu können, greift man auf die Cluster-Technologie zurück. Dabei wird ein RAID Storage-System (also ein Verbund aus Festplatten ohne dazugehörigen Rechner) mittels SCSI oder Fibre-Channel an zwei

[55] http://www.ahinc.com/raid.htm

Computer (beide natürlich wiederum auch mit RAID-System) angeschlossen. Zusätzlich verbindet man beide Systeme mit einem Netzwerk- oder Nullmodemkabel. Dieses wird Heartbeat-Monitor genannt. Im Normalfall ist nur Rechner A aktiv und bearbeitet alle Anfragen. Sollte Rechner B mittels des Heartbeat-Monitors feststellen, dass Rechner A ausgefallen ist, übernimmt er die Kontrolle über die Festplatten und die Funktionen des ausgefallenen Geräts (Abb. 2.4a). Da beide Computer sich einen Festplattenverbund teilen, spricht man hier von einem „Shared Media“ Cluster – das Gegenteil dazu wäre ein „Shared Nothing“ System. Dabei werden die Daten auf den eingebauten Speichermedien der beteiligten Rechner über eine Netzwerkverbindung synchron gehalten, es gibt daher kein externes Speichersystem. Ein solches Cluster-System werden wir im Abschnitt 5.1 im Rahmen der Verbesserungsmöglichkeiten für die DIRA-Software beschreiben.

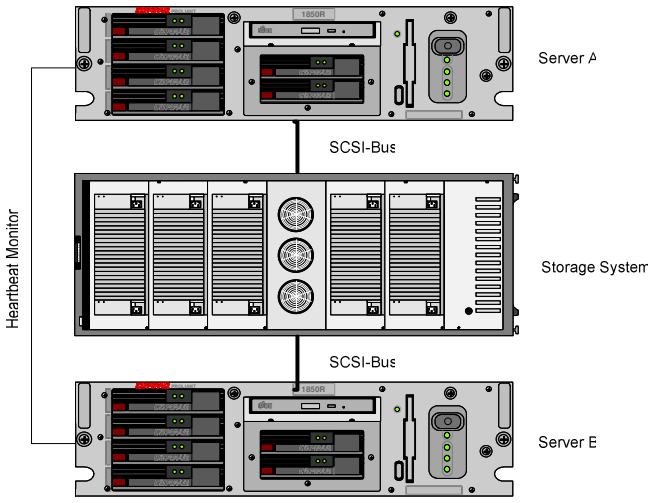

Abbildung 2.4a - Schematische Darstellung eines Shared-Media Cluster

Da das System zum Abspielen der Titel (Ausspieler) die Daten in der Regel für einige Sekunden im Voraus puffert, kann das Umschalten zwischen den beiden Computern ohne Aussetzer erfolgen[56].

[56] [SOM2002], S22ff

- Räumliche Verteilung: Selbst dieses sehr teure Cluster-System kann noch ausfallen. Dazu könnte etwa ein Feuer im Serverraum führen, oder aber auch ein Handwerker, der aus Versehen die Netzwerkkabel unterbricht. Daher ist es sinnvoll, eine Sicherung des Datenbestandes räumlich getrennt aufzubewahren.

Es ist anzumerken, dass vor allem die Clustertechnologie sehr teuer ist und sie daher in der Regel nur bei großen (und öffentlich-rechtlichen) Sendern anzutreffen ist. Aus diesem Grund begnügte man sich im Falle von Spreeradio mit mehreren Rechnern, welche im Havariefall manuell umgeschaltet werden müssen. Vor allem der Aspekt der räumlichen Verteilung ist im Zusammenhang mit der für diese Arbeit entwickelten Software sehr interessant[57]. Dabei ist allerdings anzumerken, dass diese Technologie für Hochverfügbarkeits-Systeme über eine zu niedrige Reaktionszeit verfügt, um im Hinblick auf These 1 als kompletter Ersatz für ein Cluster zu dienen.

3. Internet-Distribution

Jeder Radiosender benötigt ein Verfahren, um sein Programm zum Empfänger zu transportieren. Die nach wie vor gängigste Lösung beruht auf der Rundfunk-Übertragung mittels elektromagnetischer Wellen. Ist dagegen eine Ausstrahlung über digitale Datennetze gewünscht, so kann man dabei entweder auf Streaming oder Downloads zurückgreifen:

- Download: Die Audiodaten werden vor der Wiedergabe komplett auf den Rechner übertragen. Übertragungsfehler, wie etwa verloren gegangene Pakete, können durch Neuübertragung leicht korrigiert werden. Nachteilig an diesem Verfahren ist, dass es nicht echtzeitfähig ist, d.h. ein Radiosender

[57] [LAP1995], S. 71 - 100

könnte etwa nur das Programm der vergangenen Stunde zum Download anbieten.

- Streaming: Hierbei werden die Audiodaten kontinuierlich an die Empfänger verschickt und von diesen dekodiert und wiedergegeben. Steht bei nur einer am Transport beteiligten Station zu wenig Bandbreite zur Verfügung, so wird die Wiedergabe beim Empfänger abbrechen oder zumindest stocken.

Daher gilt es bei der Internet-Distribution, die Probleme der Bandbreite und der Echtzeitfähigkeit zu lösen. Um zu erläutern, inwieweit das im Rahmen dieser Arbeit erstellte System sich vom klassischen Streaming unterscheidet, sollen im Folgenden die aktuellen Konzepte und ihre Schwächen für die Übertragung von Audidaten im Internet beleuchtet werden.

Die Übertragung von Audio- und Bilddaten im Internet ist noch nicht zufrieden stellend gelöst. Das hat größtenteils historische Gründe: Bei der Entwicklung des ARPA-Nets (dem Vorgänger des Internets) wurde großer Wert auf die Robustheit der Netzwerkübertragung gelegt. Man wollte erreichen, dass auch beim Ausfall von einzelnen Verbindungsstrecken noch Daten zwischen Endpunkten transferiert werden konnten. Dies kann durch die paketorientierte Übertragung und individuelles, flexibles Routing erzielt werden. Dadurch ergibt sich allerdings für das Streaming eine Reihe von Problemen.

3.1. Paketorientierung

Anders als beim Telefonnetz, bei dem man nach dem Abheben des Hörers eine zugesicherte Bandbreite erhält (z.B. 64 KBit/s bei ISDN), gab es im Internet ursprünglich keine zugesicherte Verbindungskapazität. Vielmehr werden die zu transportierenden Daten in Pakete zerlegt, die dann von den beteiligten Routern durch das Netz geleitet werden (siehe Abbildung 3.1a).

Leitungs- und paketvermittelte Übertragungen lassen sich durch folgende Punkte

charakterisieren:

- Leitungsvermittelt: Einer Nachrichtenverbindung wird zeitweilig ein durchgeschalteter Übertragungskanal mit konstanter Bandbreite zugeordnet, die zur exklusiven Nutzung zur Verfügung steht. Nach Beendigung der Verbindung wird der Übertragungskanal wieder abgebaut.

- Paketvermittelt: Die beschriebenen Pakete (auch Datagramme genannt) werden ohne direkte Verbindung zwischen den beiden Endpunkten über mehrere Zwischenstationen übertragen. Daher spricht man auch von einem verbindungslosen Dienst[58]. Dabei nutzen alle Pakete eine gemeinsame Infrastruktur, die Wegewahl innerhalb dieser wird als Routing bezeichnet.

Verbindungslos bedeutet aber nicht, dass der Sender nicht weiß, ob der Empfänger die Daten auch empfängt. So ist mit Hilfe des TCP Protokolls möglich, eine gesicherte Verbindung aufbauen, bei der (im Normalfall) keine Daten verloren gehen. Ein Benutzer hat in der Regel allerdings nicht die Möglichkeit, für den Datentransport von etwa Europa nach Amerika eine feste Bandbreite zu definieren. Die Gründe dafür werden wir im Folgenden behandeln.

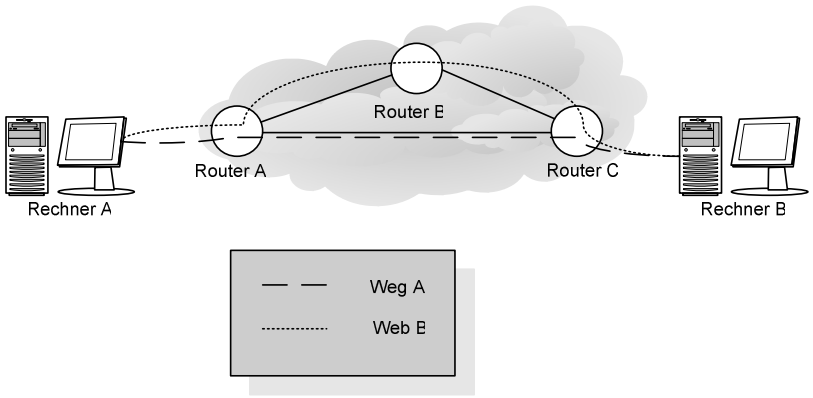

Abbildung 3.1a - Dynamisches Routing im Internet

[58] http://de.wikipedia.org/wiki/Paketvermittlung

Die beim Transport der Pakete beteiligten Router A, B, und C (Abb. 3.1a) entscheiden dynamisch, über welche Ausgänge sie die Daten weiterleiten. Im Fall von Weg A entscheidet sich Router A für den direkten Weg, bei Weg B entscheidet er sich für Router B. Dies könnte etwa der Fall sein, wenn die direkte Verbindung zu Router C gestört ist. Die Router betrachten dabei jedes ankommende Paket gleichwertig, falls ein Router ausgelastet ist, so wird der ankommende Datenverkehr in einer Warteschlange gespeichert, wodurch sich die Verarbeitung natürlich verzögert. Das sorgt für einen fehlerfreien Betrieb, ist aber für das Streaming, bei dem ein möglichst kontinuierlicher Datenstrom übertragen werden soll, sehr abträglich[59].

Um dieses Problem zu lösen, wurden beim Internet-Technologie-Konsortium IETF (Internet Engineering Task Force) verschiedene Vorschläge eingereicht, die das Internet um den Faktor Dienstgüte (oder Quality of Service) erweitern sollen.

3.1.1. Integrated Services

In den Jahren 1995 bis 1997 erarbeitete eine Forschungsgruppe des IETF verschiedene Spezifikationen, mit denen man Dienstklassen reservieren kann, daher wird auch das Kürzel RSVP (für Reservation Protocol) verwendet. Unter Dienstklassen versteht man Anforderungen an den zu übertragenen Datenstrom, zum Beispiel "mit einer maximalen Verzögerung von 100 ms" oder "mit einer minimalen Bandbreite von 500 KBit/s". Im Einzelnen handelt es sich dabei um:

- Durchsatz (Minimum/Maximum)
- Verlust
- Delay
- Jitter

RSVP ist dabei empfängerorientiert, das heißt, dass etwa der Hörer eines Internetradios dem Netz mitteilt, ihm eine gewisse Bandbreite dynamisch zu

[59] [PEL2000], S. 244 - 257

reservieren. An diesem Punkt liegt auch schon ein Problem dieses Protokolls: Wer darf Bandbreite reservieren und vor allem: Wer bezahlt dafür? Weiterhin stellt das RSVP Protokoll höhere Ansprüche an die Internet-Router, da diese die Datenpakete nicht einfach mehr nach dem „First In, First Out“ Prinzip in den Wartepuffer laden und abarbeiten dürfen, sondern schon vorher erkennen müssen, dass sie bestimmte Pakete bevorzugt behandeln müssen.

Der Hauptgrund, warum RSVP sich nicht weiter durchgesetzt hat, ist aber die mangelnde Skalierbarkeit des Protokolls. Da für jeden Datenfluss eine individuelle Reservierung benötigt wird, steigt die Belastung der Router bei einer großen Anzahl von Nutzern erheblich an[60].

Diese Gründe haben dazu geführt, dass RSVP fast ausschließlich in Forschungsnetzen wie dem MBONE verwendet wird. Neben dem Problem der Skalierbarkeit sind die Anforderungen an die Hardware zu hoch und die Abrechnung der dynamischen Reservierung zu komplex und somit zu teuer, als dass sie einer breiten Öffentlichkeit zugänglich gemacht werden können[61].

Es sei noch erwähnt, dass der Begriff Integrated Services aus dem Bereich der Telefonnetze stammt und diese Technik dort schon wesentlich länger im Einsatz ist. So startete die Post bereits 1979 ein Pilotprojekt für Integrated Services Digital Networks, kurz ISDN. In diesem Netzwerk sind Mechanismen vorhanden, mehrere ISDN-Leitung für etwa eine Audioübertragung in CD-Qualität (H1 – 1,4 MBit/s) oder eine Videokonferenz (H4 – 154 MBit/s) zusammenzuschalten. Durch die sehr hohen Kosten konnten sich diese Nutzungsmöglichkeiten von ISDN jedoch nicht durchsetzen und wurde von IP-basierten Lösungen verdrängt[62].

---

[60] http://www.tecchannel.de/netzwerk/networkworld/technologyupdate/1156/3.html

[61] [PEL2000], S. 486 - 496

[62] http://is.uni-sb.de/studium/handbuch/BISDN.php

3.1.2. Differentiated Services

Bei diesem Verfahren wird der dynamische Teil des RSVP Protokolls weggelassen und eine statische Reservierung durchgeführt. Dabei werden die Datenpakete in unterschiedliche Dienstklassen eingeteilt. So werden etwas Datenpakete mit hoher Priorität bevorzugt behandelt. Innerhalb der einzelnen Klassen erfolgt die Zustellung weiterhin nach dem Best-Effort Verfahren. Mit den Differentiated Services kann ein Provider seinen Kunden etwa einen Premium-Dienst anbieten, der dessen Daten bevorzugt transportiert. Dazu setzt der erste Router (typischerweise der des Premium-Kunden) ein oder mehrere Bits im Header der Pakete und markiert diese als „Expressgut“. Alle weiteren Router im Netzwerk des Providers erkennen die gesetzten Bits und behandeln diese ebenfalls bevorzugt. Verlassen die Pakete allerdings das Netzwerk des Providers, so funktioniert dieses Verfahren nur noch, falls dieser mit dem Peering-Partner einen entsprechenden Kooperationsvertrag abgeschlossen hat[63].

Ein weiteres Problem ist die immer noch bestehende Best-Effort Zustellung. Da die vom Betreiber eines Streams gewählte Güteklasse für alle Hörer gilt und es zudem keine Garantien für die Einhaltung dieser Güteklasse gibt, wird ein zu großer Ansturm von Nutzern weiterhin zu Störungen im Netzwerk führen.

Historisch gesehen wurden also zwei verschiedene Lösungsansätze für die Probleme der unvorhersehbaren Bandbreite und Verzögerung einer Internetverbindung entwickelt. Integrated Services wird aufgrund der schlechten Skalierbarkeit und der weiteren beschriebenen Probleme kaum angeboten. Differentiated Services bieten im Hinblick auf das Streaming von Audio- und Video keine Lösung[64]. Es wird also weiterhin nach Möglichkeiten gesucht, die beschriebenen Probleme beim Streaming zu umgehen.

---

[63] [PEL2000], S. 497 - 500

[64] [LAT1999], S. 83 - 107

3.2. Unicast und Multicast

Das nächste Problem ergibt sich aus dem Ansatz, dass das Internet als Punkt-zu-Punkt Netzwerk entworfen wurde.

Wenn etwa ein Internet-Radiosender 100 verschiedene Zuhörer hat, so müssen die Daten des Streams 100-mal an verschiedene Adressen verschickt werden, der Verkehr verhundertfacht sich also auf ausgewählten Routern! Mit den herkömmlichen Verfahren ist es nicht möglich, dass die Router mehrere Streams bündeln.

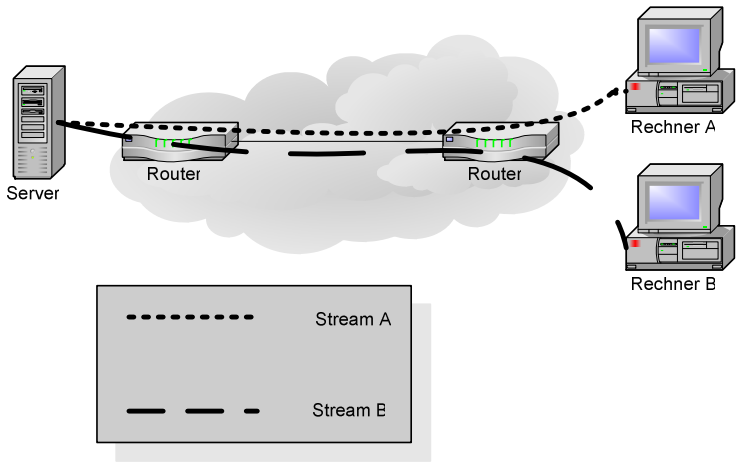

Abbildung 3.2a - Unicast Verbindungen

In Abbildung 3.2a ist ein sehr einfaches Beispiel für dieses Problem zu sehen. Obwohl Rechner A und B beide dieselben Daten erhalten (da sie z.B. einen identischen Internet-Radiosender hören), müssen sowohl der Server als auch die beiden Router die doppelte Datenmenge bewältigen.

Dadurch ergibt sich für den Betreiber des Servers eine Reihe von Nachteilen:

- Höhere Anforderungen an die Hardware: Offensichtlich belastet eine solche Übertragung die Netzwerkverbindung des Servers sowie die beteiligte

Infrastruktur sehr stark. Die Belastung steigt direkt proportional mit der Anzahl der Zuhörer an. Nimmt man für den Server eine handelsübliche 100 MBit/s Anbindung, so kann dieser zum Beispiel maximal etwa 440 Clients mit einer Bitrate von 128 KBit/s bedienen (Maximale Übertragungsrate 100 MBit-Netzwerk etwa 65-70 MBit wegen Ethernet-Overhead, Bandbreite eines 128 KBit-Streams etwa 144 KBit/s wegen Streaming-Overhead).

Will man mit Internet-Radio in die Zuhörerzahl-Bereiche der terrestrischen Radios vorstoßen, so muss man sehr viel Geld in die Technik stecken. Für etwa 100.000 gleichzeitige Hörer müsste man circa 220 Rechner über eine 22 GBit/s Netzwerkverbindung mit dem Internet verbinden. Doch nicht nur die Kosten für die Technik wären sehr hoch, auch der Datentransfer wäre extrem teuer. Der anfallende Datentransfer berechnet sich nach folgender Formel:

$$Traffic = \sum_{Hörer} (Bitrate * Hördauer_{Hörer})$$

Formel 3.2a – Benötigter Traffik beim Streaming

Es ist ersichtlich, dass beim Streaming der benötigte Traffik direkt proportional mit der Anzahl der Hörer und der Bitrate ansteigt.

- Traffikkosten: Bei dem in Abbildung 3.2a dargestellten Beispiel würde der Datenverkehr zwischen den Routern nichts kosten (vorausgesetzt sie gehören derselben Person oder Firma). Das Internet besteht allerdings nicht aus einem einzigen Netzwerk, sondern ist vielmehr eine Ansammlung verschiedenster Netze mit unterschiedlichen Eigentümern, welche über das TCP/IP Protokoll miteinander verbunden werden. Diese Netzwerke nennt man autonome Systeme. Sie alle haben zu eigen, dass sie über so genannte Peerings Daten mit anderen autonomen Netzen austauschen können[65]. Dabei muss der Partner nicht direkt erreichbar sein, sondern kann auch über ein

[65] [PEL2000], S. 304 - 311

drittes autonomes System erreicht werden. Ein solches Beispiel ist in Abbildung 3.2b dargestellt.

Im Gegensatz zum Datenverkehr im eigenen Netz, der den Betreiber wenig kostet (lediglich die Betriebs- und Mietkosten für die Technik), ist für die Benutzung der Peerings eine Benutzungsgebühr, die vom angefallenen Datentransfer abhängt, zu bezahlen.

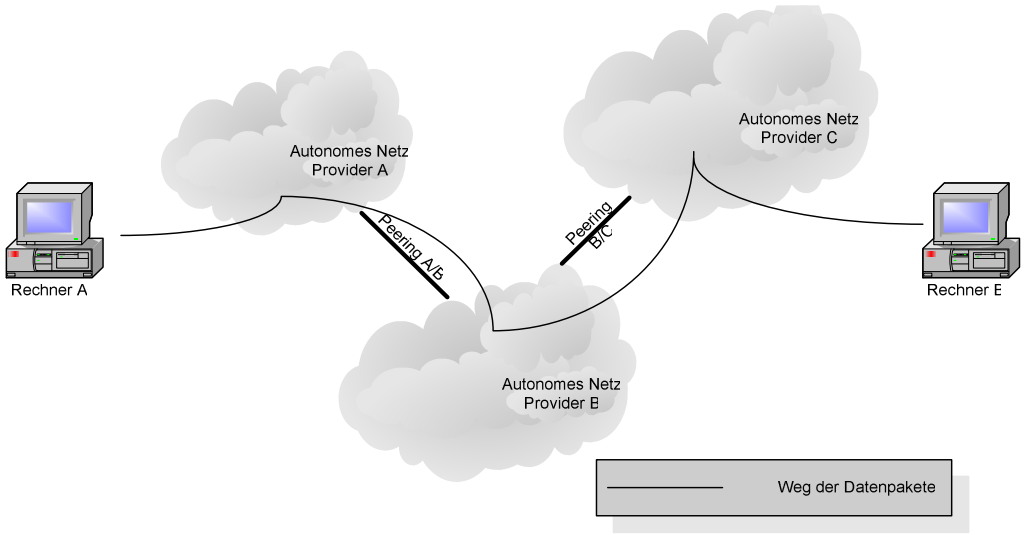

Abbildung 3.2b - Autonome Netze

Da die Provider die Gebühren für die Peerings auf die Kunden umlegen, entstehen so für den Anbieter eines Internet-Radios Kosten für den Datentransfer. Wenn etwa 1000 Leute einen Monat lang einen 128 KBit/s - Radiostream anhören, so erzeugen sie eine Datenmenge von circa 33 Terabyte, entsprechend 33.000 Gigabyte. Die günstigsten Angebote für Traffik lagen bei Erstellung dieser Arbeit bei etwa 1 Euro pro Gigabyte, ein nicht zu unterschätzender Kostenpunkt. Man muss bedenken, dass man mit dieser Zuhörerzahl noch weit von denen terrestrischer Radiosender entfernt ist und somit auch über wesentlich geringere Werbeeinnahmen verfügt. Die Finanzierung eines solchen Radios ist daher äußerst schwierig.

Zur Lösung dieses Problems wurde von der IETF ein Verfahren entwickelt, mit dem es möglich ist, ein Paket gleichzeitig an mehrere verschiedene Empfänger zu schicken. Im Unterschied zum eben beschriebenen Unicast-Verfahren wurde dieses Multicast getauft.

Dazu benutzt man einen reservierten IP-Adressbereich, bei dem unter einer bestimmten Adresse, vergleichbar mit der Frequenz eines analogen Radiosenders, ein Datenstrom abrufbar ist. Anders als bei normalen IP-Adressen sorgt das Netzwerk in diesem Fall dafür, dass möglichst viele Datenströme zusammengefasst werden. Damit ein Zuhörer einen Multicast-Radiostream hören kann, so muss er seinen Router anweisen, der gewünschten Multicast-Gruppe (welche in der Regel identisch mit einer Multicast-Adresse ist) beizutreten. Schematisch lässt sich dies folgendermaßen darstellen:

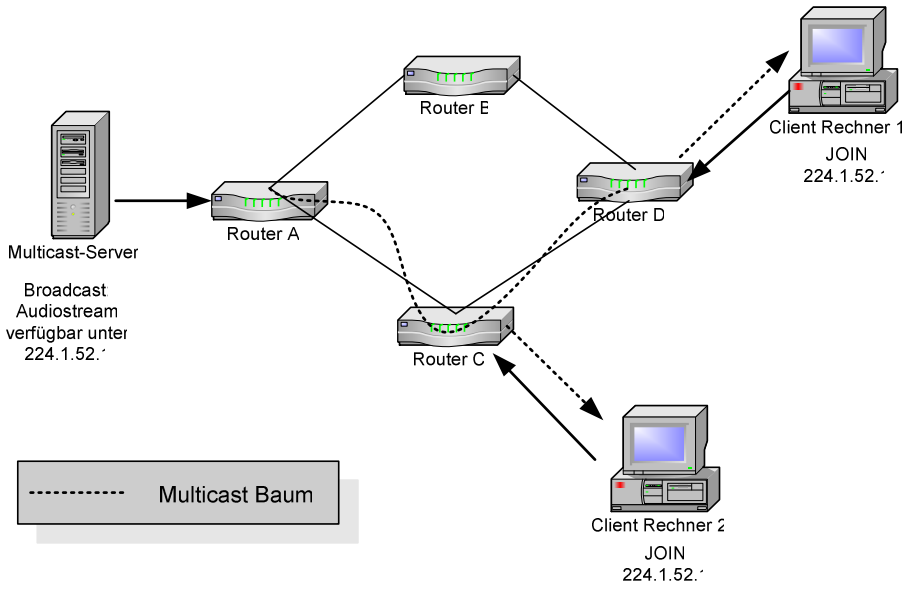

Abbildung 3.2c - Multicast Baum

Der Initiator eines Streams gibt zum Beispiel über eine Webseite oder andere Out-of-band Mechanismen bekannt, dass der Datenstrom unter der IP-Adresse 224.1.51.1 verfügbar ist. Der Rendevous-Punkt ist in diesem Fall Router A. Wollen Client eins und zwei diesen Strom erhalten, so teilen sie "ihrem" ersten Router (zum

Beispiel dem ihrer Universität) mittels des Internet Group Message Protocol (IGMP) mit, dass sie der Multicastgruppe 224.1.51.1 beitreten wollen (JOIN). Router A, C und D handeln daraufhin einen Baum zur Bündelung des Datenstroms aus. Die Übertragung der Daten zwischen Router und Client-Rechner findet dabei wieder im Unicast-Verfahren statt. Es existieren auch andere Multicast-Varianten, bei denen die Daten durch den ersten Router in das Netz geflutet werden, und die weiteren Router den Stream bei diesem abbestellen, wenn sie in ihrem Subnetz keine Empfänger ausmachen können (PRUNE). Dieses Verfahren eignet sich gut für dichte Netzstrukturen wie etwa MANs (Metropolitan Area Network) und wird daher auch „Dense Mode“ bezeichnet.

Die größte Herausforderung bei Multicast ist es, einen geeigneten Baum für die Übermittlung zu erstellen. Dabei darf man nicht von dem Beispiel in Abbildung 3.2c ausgehen, sondern von einem wesentlich komplexeren Netz mit mehreren hundert Routern und autonomen Systemen. Weiterhin soll gewährleistet sein, dass bei Ausfall eines Routers innerhalb kürzester Zeit ein Alternativ-Baum aufgebaut wird. Die dafür verwendeten Algorithmen sind sehr komplex und würden den Umfang dieser Arbeit sprengen. Am weitesten fortgeschritten ist dabei das PIM Protokoll (Protocol Independent Multicast), welches neben der Plattformunabhängigkeit auch verschiedene Mechanismen zur Vermeidung von unnötigem Traffik und Zusammenfassung unterschiedlicher Multicastgruppen bietet[66,67,68].

Kehrt man allerdings in den Internet-Alltag zurück, so stellt man fest, dass Multicast ein absolutes Exotendasein fristet. So gut wie kein kommerzieller Provider in Deutschland erlaubt seinen Kunden die Benutzung der Multicast-Funktionen seiner Router. Das liegt daran, dass Multicast (im Vergleich zum normalen IP-Traffik) sehr kompliziert ist und erst aktuelle Router es einwandfrei beherrschen. Da mangels entsprechender Multicast-Angebote auch keine große Nachfrage nach diesem Verfahren herrscht, scheuen viele Provider die Kosten für die Umrüstung ihrer

---

[66] http://www.ietf.org/html.charters/pim-charter.html

[67] [PEL200], S.336 – 344

[68] [COD2000], S. 319 - 349

Hardware. Daher muss man leider sagen, dass Multicast außerhalb einiger Uni-Netzwerke wie dem DFN im Internet kaum eine Rolle spielt.

3.3. Distributed Computing

Auf Grund der stark gefallenen Hardwarepreise und der immer schneller werdenden Netzwerke hat sich das Systemdesign von Hochleistungsrechnern (oder besser Rechneranlagen) stark verändert. Setzte man früher für die Bewältigung großer Datenmengen auf wenige, sehr teure Mainframes, so ist man heute dazu übergegangen, viele günstige Standardrechner zu einem Verbund zusammenzuschalten. Die Vorteile sind neben dem wesentlich günstigeren Preis die höhere Ausfallsicherheit, der Ausfall eines Rechners kann durch die vielen anderen leicht kompensiert werden.

In Bereichen, in denen riesige Datenmengen anfallen (z.B. Klimaforschung, Atomforschung), werden diese häufig von so genannten Cluster Computern verarbeitet. Die Grenzen zwischen Cluster Computing und Grid Computing sind eher fließend, in der Regel bezeichnet man einen Rechnerverbund als Cluster, wenn alle beteiligten Maschinen über die gleiche Hardware verfügen, fest über Realzeit-Netze gekoppelt sind und dediziert für diesen Verbund arbeiten[69]. Auf der Liste der weltweit schnellsten Rechner[70] befindet sich das erste Mainframe System an Stelle 19 (Cray X1 mit 256 Prozessoren), die Plätze 1 - 18 sind von Cluster Systemen belegt. Der Begriff Cluster ist in diesem Zusammenhang etwas irreführend, er bezeichnet hier nicht ein System zur Erhöhung der Ausfallsicherheit, sondern einen Verbund aus sehr vielen Rechnern. Interessant ist noch, dass sich dieses Jahr das Los Alamos Labor[71] mit einem selbstgebauten Verbund aus 1100 Dual G5-Macs, verbunden durch normales Gigabit-Ethernet, den dritten Platz sichern konnte. Die Kosten für das System betrugen 5,2 Millionen US-$, ein Bruchteil des (allerdings

[69] http://en.wikipedia.org/wiki/Distributed_computing

[70] http://www.top500.org/list/2003/11/

[71] http://www.lanl.gov/

auch knapp 3-mal so schnellen - er besitzt 5120 Prozessoren) Erdbebensimulator in Japan, der etwa 350 Millionen US-$ kostete[72,73].

Aber nicht nur im Bereich Großrechenanlagen hat sich das verteilte Rechnen durchgesetzt. Die sehr populäre Suchmaschine Google setzt zum Beispiel ebenfalls auf etwa 10.000 weltweit aufgestellte Server[74] aus Standard-Hardware. Durch ein sehr ausgefeiltes Load-Balancing System können enorm kurze Antwortzeiten erzielt werden. So dauerte die Suche nach obigem Supercomputer etwa eine viertel Sekunde, bei 3 Milliarden gespeicherten Webseiten ein beachtlicher Wert[75,76].

Nachteilig an diesen Systemen ist, dass für die Anschaffung der Hardware eine beträchtliche Summe investiert werden muss. So war es eine Frage der Zeit, bis die Idee, einzelne Rechner mittels einer Software über das Internet zu einem Rechnerverbund zusammenzuschließen, geboren wurde. Das erste System dieser Art war das Mersenne Prime Search Projekt, welches 1996 gestartet wurde. Teilnehmer können sich eine Software auf ihrem Rechner installieren, welche ungenutzte Rechenleistung zur Suche nach unbekannten Mersenne'schen Primzahlen benutzt[77]. Diese Software-Clustersysteme werden als Grid bezeichnet[78].

Wesentlich mehr Teilnehmer konnte das Seti@Home Projekt[79] gewinnen. Das "Search for extraterestial life" Projekt der Universität von Berkeley sucht mittels Radioteleskopen nach außerirdischem Leben und erzeugt dabei eine Datenmenge von etwa einem halben Megabyte pro Sekunde, welche an die verbundenen Clients verteilt wird. Die Teilnehmer installieren eine Software auf ihrem Rechner, welche kleine Datenpakte (etwa 350 Kilobyte) vom Server des Projekts erhält. Nachdem der Rechner des Teilnehmers die Daten analysiert hat (was etwa einen halben Tag

---

[72] http://www.technews.vt.edu/Archives/2003/Nov/03665.htm

[73] [ZOA1996], S. 762 - 779

[74] http://dance.kunden-efactory.de/statistik/stats.php

[75] http://www.google.com/press/funfacts.html

[76] [KAS2003], S. 88-90

[77] http://www.cushat.co.uk/prime/

[78] http://en.wikipedia.org/wiki/Grid_computing

[79] http://setiathome.ssl.berkeley.edu

dauert), sendet dieser die Ergebnisse zurück an das Seti-Institut. Vor allem wegen der ansprechenden Aufmachung der Client-Software (mit grafischer Darstellung der Signale und Bearbeitungsschritte) und einer Wertungsliste für bearbeitete Pakete wurde das System ein großer Erfolg. Im Juli 2002 war die Client-Software bereits 3,8 Millionen mal abgerufen worden, die erzielte Rechenleistung lag im Schnitt bei 27 Billionen Fließkommaoperationen pro Sekunde - der oben beschriebene Apple-Cluster bringt es auf gerade einmal 10 Billionen Operationen[80,81].

Diese Art des internet-gestützten Distributed Computing funktioniert allerdings nicht für alle Arten von Anwendungen. Die eben beschriebenen haben den Vorteil, dass sich die Daten sehr leicht aufteilen lassen, zwischen den einzelnen Clients keine Kommunikation notwendig ist und die zu übertragende Datenmengen relativ gering sind. Für Anwendungen, in denen etwa das Ergebnis eines Rechners die Berechnungen der anderen Clients verändert, ist ein solches System nicht geeignet, da dann eine intensive Kommunikation zwischen dem Zentralrechner und den Clients geführt werden muss[82].

Für das im Rahmen dieser Arbeit entwickelte Distributionssystem steht allerdings weniger die Verteilung von hoher Rechenleistung auf mehrere Systeme im Vordergrund, sondern vielmehr die Erhöhung der Betriebsstabilität bei Verwendung von unzuverlässigen, nichtpermanenten Netzwerkverbindungen. Auf diese Weise sollen die beschriebenen Unzulänglichkeiten des Internet-Streamings umgangen werden.

3.4. Peer-to-peer Netze

Die in Abschnitt 3.3 beschriebenen Cluster- und Grid-Systeme haben alle gemeinsam, dass ein zentraler Rechner für das Verteilen (und Einsammeln) der Daten sowie für die Steuerung des Systems zuständig ist. Die Clients-Software ist

---

[80] http://setiathome.ssl.berkeley.edu/cacm/cacm.html

[81] http://www.bio-itworld.com/news/111703_report3789.html

[82] [ZOA1996], S. 725 - 759

dabei relativ einfach ausgeführt, sie tätigt nur die Berechnungen und weiß in der Regel nicht, was andere Clients parallel machen. Da der Zentralserver der Angelpunkt dieser Technologie ist, bezeichnet man solche Systeme auch als monolithisch oder zentralistisch. Fällt dieser zentrale Rechner aus, so wird dadurch das ganze System lahm gelegt, da die Clients nicht in der Lage sind, selbstständig untereinander Kontakt aufzunehmen.

Mit Peer-to-peer bezeichnet man im Gegensatz dazu die Kommunikation der einzelnen Clients untereinander. Das Fido-Net, welches 1984 von Tom Jennings entwickelt wurde, war eines der ersten Systeme, welches die Peer-to-peer Technologie einsetzte.

Zu dieser Zeit waren Internet-Standleitungen noch nicht üblich, selbst Universitäten tauschten ihre Daten mittels Modemverbindungen aus. Wollte man etwa eine Nachricht an ein elektronisches, schwarzes Brett heften (Bulletin Board System), so wählte man mit seinem Modem die Nummer des entsprechenden Systems an, schrieb die Nachricht und trennte darauf die Verbindung. Dadurch entstanden bei Ferngesprächen hohe Telefongebühren.

Jennings entwickelte ein System, mit dem es möglich war, eine Nachricht bei einem beliebigen Bulletin Board System (auf deutsch auch häufig Mailbox bezeichnet – Anrufbeantworter wurden erst zu einem späteren Zeitpunkt zu Mail- oder Voicebox umbenannt) zu hinterlassen, welche automatisch an alle weiteren Boards geleitet wurde. Um Telefongebühren zu sparen, wurde diese Weiterleitung meist nachts durchgeführt, Laufzeiten von mehreren Tagen waren die Regel.

Interessant an dieser Anwendung ist aber, dass schon damals viele Problemfelder wie Routing, Skalierbarkeit und Sicherheit entdeckt und teilweise gelöst wurden. Allerdings fehlten zu dieser Zeit die Rechenleistung und die Softwaretechnik, als dass man diese Probleme (die auch heute teilweise noch ungelöst sind) vollständig in den Griff bekommen hätte.[83,84]

[83] http://www.ibm.com/developerworks/java/library/j-p2p/

[84] http://www.wps.com/FidoNet/

Die erste äußerst erfolgreiche Peer-to-peer Software entwickelte 1999 der damals 19-jährige Shaun Fenning mit dem Napster-System. Dieses bot ein Client-Programm, welches es ermöglichte, Musikstücke zwischen den Benutzern (welche alle die Client-Software ausführen müssen) des Systems auszutauschen. Der zentrale Server diente dabei lediglich der Koordinierung der angeschlossenen Clients und der Suchanfragen, die voluminösen Datentransfers hingegen fanden direkt zwischen den Clients, also peer-to-peer, statt. Auf diese Weise hielt sich die Last auf den Servern des Betreibers in Grenzen. Da zu dieser Zeit Breitbandzugänge für Privatanwender immer populärer wurden, konnten die Teilnehmer ihren Peer-to-peer Tauschpartner genug Kapazitäten für schnelle Transfers zur Verfügung stellen.

Laut eigener Aussage hatte die Tauschbörse bis zu 30 Millionen Benutzer weltweit. Als Napster dann im September 2001 auf einen Gerichtsbeschluss hin seinen Betrieb einstellen musste, gab es bereits eine Vielzahl weiterer Tauschbörsen, die nach einem vergleichbaren Konzept arbeiten. In den letzten Jahren sind auch etliche Plattformen für den Tausch von Filmen auf den Markt gekommen - mit Technologien wie DSL ist eine ausreichend schnelle Internetverbindung verfügbar[85].

Die Abschaltung von Napster war relativ einfach, da diese Software, wie auch Seti@Home, eine zentralisierte Topologie verwendete. Das erste System, welches eine rein dezentralisierte Topologie verwendete, war das Gnutella-Netzwerk, für welches eine ganze Reihe von Clients existiert[86]. Es gibt bei dezentralen Peer-to-peer Netzen allerdings eine Reihe von Problemen, wie in folgender Grafik deutlich wird.

[85] http://web.utk.edu/~smarcus/History.html

[86] http://en.wikipedia.org/wiki/Gnutella

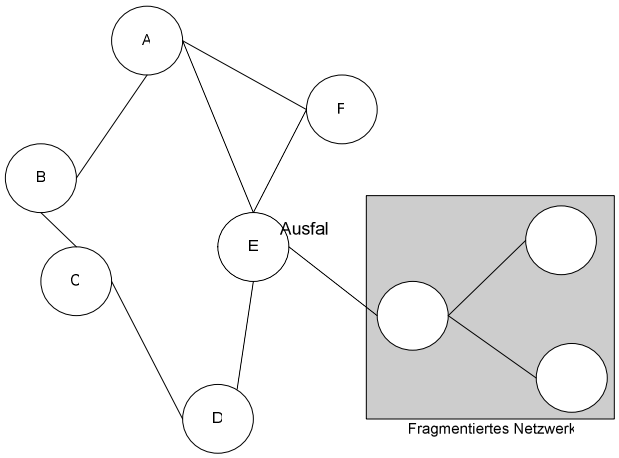

Abbildung 3.4a – Ausfall eines Nodes in dezentralen Netzwerken

Da die Teilnehmer dieses Netzes (auch Nodes genannt) meist nicht über eine feste IP-Adresse verfügen oder weil ein Anwender die Client-Software beendet, kommt es häufig vor, dass einzelne Nodes ausfallen. Das führt zu zwei Problemen:

- Fragmentierung: Wie in der Grafik deutlich wird, verliert der grau unterlegte Bereich bei Ausfall des mittleren Nodes die Verbindung zum Netzwerk. Die Rechner bilden daraufhin ein eigenes Netzwerk, welches keinen Zugang zu den Rechnern des linken Netzwerks hat.
- Routing: Alle Nodes müssen sehr häufig ihre Routing-Informationen ändern. Hat Node D etwa gelernt, dass er F über E erreichen kann, so muss er die neue Route über C, B und A erst lernen, was durchaus einige Zeit in Anspruch nehmen kann. Während dieser Zeit können Suchanfragen verloren gehen.

Diese beiden Problemfelder sind Thema etlicher Forschungsarbeiten. Es ist möglich, dass diese Unzulänglichkeiten durch verbesserte Algorithmen zu lösen sind[87,88].

[87] http://www.openp2p.com/pub/a/p2p/2001/12/14/topologies_one.html

[88] http://www.openp2p.com/pub/a/p2p/2002/01/08/p2p_topologies_pt2.html

Die zurzeit populärste Tauschbörse Kaazaa[89] (im Schnitt 3 Millionen gleichzeitige Teilnehmer!) benutzt daher ein hybrides Verfahren. Nodes, die über eine schnelle Verbindung und eine gute Erreichbarkeit verfügen, werden zu so genannten Supernodes befördert. Die Supernodes bilden ein dezentrales Netzwerk, die Clients sind normalerweise mit nur einem Supernode verbunden. Da sie die Adressen der weiteren Supernodes aber übermittelt bekommen, können sie bei Ausfall „ihres" Supernodes auf einen anderen wechseln. Vorteil dieses Verfahren ist, dass sich die Routing-Informationen nur selten ändern. Ein ähnliches Verfahren wird im Edonkey Netzwerk[90] eingesetzt – hier wird die Client-Software allerdings nicht automatisch zu einem Supernode befördert, sondern es muss ein spezielles Server-Programm ausgeführt werden.

Es gibt aber neben den meist illegalen Angeboten in den erwähnten Tauschbörsen auch legale Anwendungszwecke für die Peer-to-peer Technologie: So wird zum Beispiel die Software *BitTorrent* von etlichen Linux-Distributoren genutzt, um die sehr voluminösen DVD-Images des Betriebssystems (um die 4,5 GByte) im Internet zum Download anzubieten[91].

Betrachtet man aktuelle Statistiken über die Art des auftretenden Internet-Datenverkehrs, so stellt man fest, dass dieser teilweise zu 70 % aus Peer-To-peer Übertragungen besteht[92]. Dadurch wird deutlich, dass der Einsatz dieser Technologie ein großer Erfolg ist und diese bereits ausgiebig genutzt wird.

In der derzeitigen Version ist für das DIRA-System nur der Austausch mit einem zentralen Server vorgesehen. Es ist allerdings technisch relativ leicht möglich, einen Peer-to-peer Austausch unter den Clients zu ermöglichen, was die Skalierbarkeit der Software nochmals enorm erhöhen würde. In Abs. 5.3 wird genauer auf die benötigten Anpassungen eingegangen.

---

[89] http://www.kazaa.com/us/index.htm

[90] http://www.edonkey2000.com/

[91] http://www.debian.org/CD/torrent-cd/

[92] http://www.cachelogic.com/research/slide3.php

3.5. Zusammenfassung Internet-Streaming

Da die großen Carrier am Datentransfer durch ihr Netzwerk verdienen, haben diese wenig Interesse, Verfahren wie Multicast, welche ja die Datenmenge verringern sollen, einzuführen. Hier wird vielmehr versucht, die angesprochenen Probleme durch mehr Bandbreite zu lösen. Da die Übertragungsgeschwindigkeiten der Netzwerke immer höher und diese zugleich kostengünstiger werden, gelingt dies noch sehr gut. Will man aber das Internet zur oft prophezeiten multimedialen, on-demand-Videothek ausbauen, noch dazu mit der gleichen Zuverlässigkeit wie etwa das bestehende Kabelfernsehen, so wird man nicht umhin kommen, Technologien wie Multicast und RSVP einzusetzen. Eine Internet-Übertragung per Unicast etwa der Tagesschau mit 5 Millionen gleichzeitigen Zuschauern ist zurzeit technisch nicht realisierbar. Bei guter Bildqualität (500 KBit/s) müsste hier eine Bandbreite von über 2,3 Terabit zu Verfügung haben stehen[93]. Dies gilt natürlich umso mehr für globale Veranstaltung wie etwa die Fußball-Weltmeisterschaft mit weltweit rund 1 Milliarde gleichzeitigen Zuschauern[94].

Verbleibt man dagegen in einem kleineren Rahmen, so bietet Internet-Streaming schon jetzt, trotz der erwähnten Schwierigkeiten, eine sehr interessante Möglichkeit, einen eigenen Radiosender zu gestalten. Auf Seiten wie *Shoutcast*[95] oder *Icecast*[96] finden sich über 10000 verschiedene, kostenlose Radiostationen. Diese werden meistens von Privatpersonen betrieben, die aus Interesse an einer bestimmten Musikrichtung die Kosten für die Übertragung in Kauf nehmen[97].

---

[93] http://www.aes.org/technical/documents/i2.pdf

[94] http://www.fifa.com/de/marketing/newmedia/index/0,1347,10,00.html

[95] http://www.shoutcast.com

[96] http://yp.icecast.com

[97] [AKB1994], S. 21 - 41

4. Das DIRA-System

Das im Rahmen dieser Arbeit entwickelte DIRA-System soll die technologische Nachfolge des TOX Internet-Radiosystems antreten und durch den kombinierten Einsatz verschiedener Übertragungsmethoden dessen Skalierbarkeit und Robustheit nachhaltig verbessern. Im Folgenden soll daher eine Anforderungsanalyse für ein verteiltes Internet-Radiosystem erstellt werden.

4.1. Anforderungen

Für die Wiedergabe der Audiodaten durch die Clients sind folgende Punkte von Bedeutung:

- Gleichbleibende Lautstärke der Titel: Es dürfen keine hörbaren Lautstärkeschwankungen auftreten.
- Blenden zwischen den Elementen: Um einen flüssigen Ablauf des Programms zu gewährleisten, sind Überblendungen zumindest bei den Musiktiteln notwendig.
- Robustheit gegenüber Störungen: Auf Grund der verteilten Struktur des DIRA-Systems müssen die Clients in der Lage sein, selbstständig Fehler zu erkennen und zu beheben.
- Havarie-Mechanismen: Im Falle einer Störung der Datenbank oder der Internetverbindung darf die Wiedergabe nicht abbrechen, es wird dann ein alternatives Programm ausgegeben.

Für den Datentransfer zwischen Client und Server sollen folgende Anforderungen erfüllt sein:

- Kein Streaming: Die Übertragung der benötigten Audiodateien soll nicht per Streaming erfolgen.

- Nutzung von Peer-to-peer-Technologie: Die Audiodateien sollen mittels der Peer-to-peer-Technologie unter den Clients ausgetauscht werden.
- Nutzung von Pull-Technologien: Die Clients sollen die benötigten Titel mittels des Pull-Verfahrens vom Server abholen.
- Nutzung von Multicast: Falls möglich sollen neue Elemente per Multicast an die Clients gesendet werden.
- Weniger Transfervolumen: Die übertragene Datenmenge soll effektiv geringer als beim Streaming-Verfahren sein

Um dem Anwender den Umstieg von einem herkömmlichen Radio-Automationsprogramm zu einem verteilten System zu erleichtern, sollte sich die Bedienung an bestehender Software orientieren. Dazu zählen:

- Programmplanung: Bietet die Möglichkeit, den Ablauf des Programms individuell zu erstellen.
- Archivverwaltung: Ermöglicht das Einfügen, Verändern und Löschen von neuen Musikstücken und Elementen in das Programm.
- Reporting: Stellt Statistiken und Berichte über die wiedergegebenen Titel bereit.

Weiterhin sind bezüglich des Datenbestandes folgende Anforderungen zu erfüllen:

- Integrität der Daten: Der Datenbestand der Clients darf sich nur kurzzeitig von dem des Servers unterscheiden. Dies ist über entsprechende Synchronisationsalgorithmen sicherzustellen.
- Benutzerauthentifizierung: Der Zugriff auf das System muss durch Passwörter und verschiedene Berechtigungsstufen abgesichert werden.
- Schutz vor Fremdzugriff: Es ist sicherzustellen, dass die Daten eines Anwender nur durch diesen verändert werden können. Dies schließt sowohl böswillige Angriffe durch etwa Hacker als auch weitere, parallel arbeitende

Benutzer ein.

Da es sich bei dem DIRA-System um eine komplexe Software sowohl zur netzwerkbasierten Übertragung von Audiodaten als auch zur Ablaufsteuerung eines Radioprogramms handelt, müssen verschiedene Anforderungen an die verwendete Hard- und Software erfüllt werden. Diese sind allerdings nicht sehr hoch und lassen sich mit geringen finanziellen Mitteln realisieren:

- Prozessorleistung ab 800 MHz (bei x86-Architektur)
- Arbeitsspeicher mit mindestens 128 (Client) bzw. 256 MByte (Server) Kapazität
- Nichtflüchtiger Datenträger mit mindestens 1 GByte Kapazität (abhängig von der Anzahl der verwendeten Titel)
- Digital-Analog Wandler zur Wiedergabe des Audiosignals
- Netzwerkinterface zu Anbindung an ein LAN oder WAN

Das DIRA-System wurde für die Linux-Systemarchitektur entwickelt. Es lässt sich aber vermutlich auf eine Vielzahl von ähnlichen Umgebungen (BSD-Varianten, Unix, HP-UX etc.) portieren. Vorausgesetzt wird dabei:

- Aktueller Kernel ab Version 2.4
- Unterstützung des TCP/IP Stacks
- Unterstützung von Sockets
- Audio-Wiedergabe durch die Module OSS oder ALSA
- Relationale Datenbank

Zusätzlich ist es erforderlich, dass das Einsatzfeld die Nutzung der General Public Licence bzw. vergleichbarer Lizenzen erlaubt.

Für die Interaktion mit dem System sind weiterhin ein Webbrowser mit Frame-

Unterstützung sowie eine MP3-Kodierungssoftware zum Erstellen der Titel erforderlich.

Diese Anforderungen an die Software sind alle durch frei im Internet erhältliche Programme abgedeckt.

4.2. Spezifikation

Das im Rahmen dieser Arbeit entwickelte DIRA-System versucht, die in Abs. 3 beschriebenen Unzulänglichkeiten der Internet-Audioübertragung mittels Streaming durch eine Kombination verschiedener Techniken und Strategien zu umgehen.

Dabei beruht das System in der Grundausstattung auf einem Server und einem oder mehreren Clients. Diese beiden Komponenten haben folgende Aufgaben:

Server:

- Die Benutzerinteraktion findet ausschließlich über ein Web-Interface statt.
- Die Programmplanung und das Einspielen neuer Titel oder Programmelemente erfolgt allein auf dem Server.
- Der Server speichert die vorhandenen Titel in Form von Audiodateien auf einem lokalen Datenträger. Die Metadaten hingegen werden in eine relationale Datenbank eingetragen.
- Die Kommunikation mit den Clients erfolgt durch eine Transfer- und eine Synchronisationsschnittstelle.

Clients:

- Die Clients geben die durch die Playlist-Verwaltung festgelegten Titel über den vorhandenen Digital-Analog-Wandler wieder.
- Die zu spielenden Titel befinden sich im Idealfall auf dem lokalen Datenträger. Es ist Aufgabe des Clients dafür zu sorgen, dass der lokale

Datenbestand identisch mit dem des Servers ist.

- Die Synchronisation des Datenbestandes soll durch folgende Transfermethoden erfolgen

    1. Übertragung per Multicast, falls verfügbar
    2. Peer-to-peer-Übertragung zwischen den Clients
    3. Individueller Serverzugriff nach dem Pull-Verfahren
    4. Streaming

- Schlägt eine Methode fehl, so soll jeweils auf die nächste ausgewichen werden.
- Schlagen alle vier Methoden fehl, so soll der Client eine alternative Datei wiedergeben
- Der Client soll statistische Daten (etwa zur Anzahl der Wiederholungen einzelner Stücke) an den Server übertragen

4.3. Analyse bestehender Systeme

Wie schon beim TOX-System soll die Steuerung der Wiedergabe ausschließlich über Webseiten geschehen. Dabei ist es sinnvoll, die Bedienung an bereits existierende Radio-Automationssysteme anzugleichen, um dem erfahrenen Anwender den Umstieg zu erleichtern.

Betrachtet man verschiedene Automationssysteme, so kann man identische Bearbeitungs- und Planungsmodule wieder erkennen. Die Umsetzung dieser Module ist natürlich von Programm zu Programm unterschiedlich.

- Playlisten-Modul: Dieses Modul dient der Steuerung des Programmablaufs. Es ermöglicht dem Benutzer (in der Regel dem für das Programm verantwortlichen Chef vom Dienst (CVD)), den Ablauf der einzelnen

Elemente zu kontrollieren. Die Planung erfolgt anhand einer sog. Programmuhr, bei der eine Stunde in verschiedene Programmsegmente unterteilt ist.

- Archiv-Modul: Der vorhandene Bestand an Musiktiteln und anderen Elementen muss gewartet werden. Zu diesem Zweck stehen in diesem Modul Such- und Bearbeitungsfunktionen zur Verfügung. Ein Beispiel für die Funktionalität dieses Moduls wäre etwa, dass die im Dezember sehr häufig gespielten Weihnachtslieder nach dem 24. nicht etwa aus dem System gelöscht, sondern nur für den Rest des Jahres deaktiviert werden und in der nächsten Saison wieder zur Verfügung stehen.
- Bearbeitungs-Modul: Hiermit können vorhandene sowie neue Titel bearbeitet werden. Zu diesem Zweck wird eine Amplitudendarstellung der Audiodatei angezeigt, in welcher mittels der Maus verschiedene Parameter angepasst werden können. Dazu zählen etwa Start- und Endpunkte sowie die Zeiträume für die Überblendungen von Elementen. Weiterhin findet hier bei neu eingespielten Titeln die Umwandlung in das benutzte Audioformat sowie eine Normalisierung der Lautstärke statt.

Diese Aufzählung stellt natürlich nur einen sehr groben Überblick über die Funktionalität professioneller Radiosoftware dar. Für die Gestaltung der Benutzeroberfläche der DIRA-Software gilt es, diese Module durch verschiedene Webseiten entsprechend abzubilden.

Durch die verteilte Systemstruktur des DIRA-Systems werden zwei weitere Module benötigt, für die keine Äquivalente in herkömmlicher Radio-Software existieren. Dies sind:

- Synchronisations-Modul: Der Datenabgleich von Client- und Server soll zeitgesteuert erfolgen. Dabei stellt der Client automatisch die Verbindung zum Server her und initiiert den Austausch von Metadaten und Audiodateien.

- Erweitertes Playlisten-Modul: Anders als bei üblichen Radio-Systemen bietet die verteilte Architektur die Möglichkeit, das Programm für jeden angeschlossenen Client individuell zu gestalten.

4.4. Entwurf

Um die Vorteile eines verteilten Radiosystems ausnutzen zu können, ist die Art des zu sendenden Radioprogramms einzugrenzen. Für das folgende DIRA-System wird davon ausgegangen, dass eine begrenzte Anzahl von Titeln vorhanden ist und dass keine Live-Schaltungen getätigt werden. Der zweite Punkt mag wie eine große Einschränkung erscheinen, in der Praxis sind solche Schaltung bei terrestrischen Radiosendern äußerst selten, da sie teuer sind und es für den Zuhörer in den meisten Fällen nicht ersichtlich ist, ob etwa ein Interview live geführt wird oder lediglich eine Aufzeichnung abgespielt wird.

Ziel dieser Arbeit ist es, eine neuartige Methode zur Übertragung eines Radioprogramms über das Internet, welche ohne das Streaming-Verfahren auskommt, zu entwickeln. In Abb. 4.4a ist beispielhaft die Arbeitsweise der TOX Automationssoftware dargestellt.

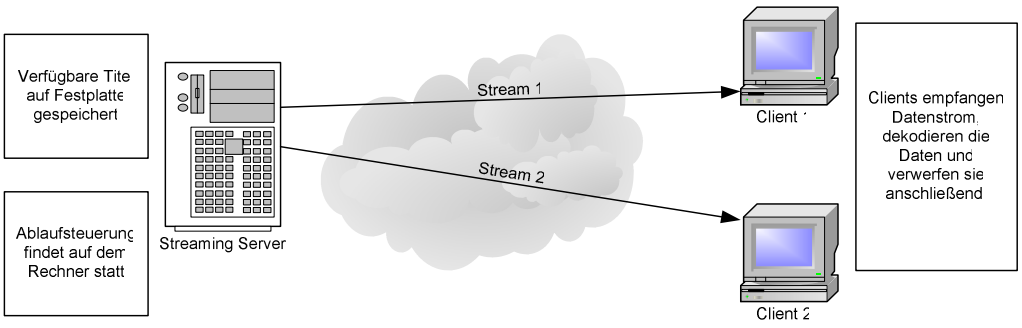

Abbildung 4.4a – Internet-Radio mit dem TOX-System

Die zu spielenden Musikstücke, Beiträge, Jingles sowie Werbeblöcke befinden sich

in MP3-kodierter Form auf dem Streaming Server. Dieser erstellt für das Programm einer Stunde eine Playlist und spielt sukzessive die gewählten Titel ab. Diese werden dabei allerdings nicht als Audio-Signal ausgegeben sondern als Datenstrom an die Clients übertragen. Diese dekodieren den Strom und geben ihn als Audiosignal wieder. Danach werden die Daten verworfen. Diese Arbeitsweise findet sich nicht nur beim TOX-System sondern bei jeglicher Audio- und Videoübertragung per Streaming.

In Unterscheidung hierzu wurden für das DIRA-System folgende Veränderungen am TOX System getätigt:

1. Um die Betriebsstabilität und die Robustheit gegenüber Störungen zu verbessern, ist es erforderlich, sich vom Konzept des Streamings zu lösen. Eine netzwerkunabhängige Wiedergabe durch den Ausspieler ist nur gewährleistet, wenn dieser zumindest über eine Teilmenge der auf dem zentralen Server gespeicherten Audiodateien verfügt.

   Aus diesem Grund ist das TOX System dahingehend zu verändern, dass der Server nicht mehr der Wiedergabe sondern nur noch der Archivierung der Dateien und der Koordination der angeschlossenen Ausspieler dient. Diese hingegen benötigen einen nicht-flüchtigen Datenträger (Festplatte oder Ähnliches), um die vorgesehenen Titel lokal zu speichern. Dazu müssen sie über eine Möglichkeit zur Übertragung von Daten vom Server verfügen. Weiterhin wird die Rechenarbeit zur Erstellung des Programms auf die Clients ausgelagert.

2. Da das Programm nicht statisch sein soll, sind Client und Server über geeignete Synchronisationsalgorithmen abzugleichen. Dies beinhaltet sowohl die in Punkt 1 beschriebenen Transfers der Audiodaten als auch die zur Erstellung eines Programms benötigten Metadaten.

Dieser Abgleich findet periodisch zu festgelegten Zeitpunkten statt. Dadurch wird die Stabilität des Systems verbessert und gleichzeitig die Menge der durch den Server zu transportierenden Daten erheblich gesenkt.

3. Verbessert man das Playlisten-Modul dahingehend, dass für jeden angeschlossenen Client ein eigenes Programm planbar ist, so erreicht man einen wesentlich höheren Grad der Individualisierbarkeit der Programmgestaltung als bei herkömmlichen Radiosystemen.

Dieses verteilte Audiosystem würde sich schematisch folgendermaßen darstellen:

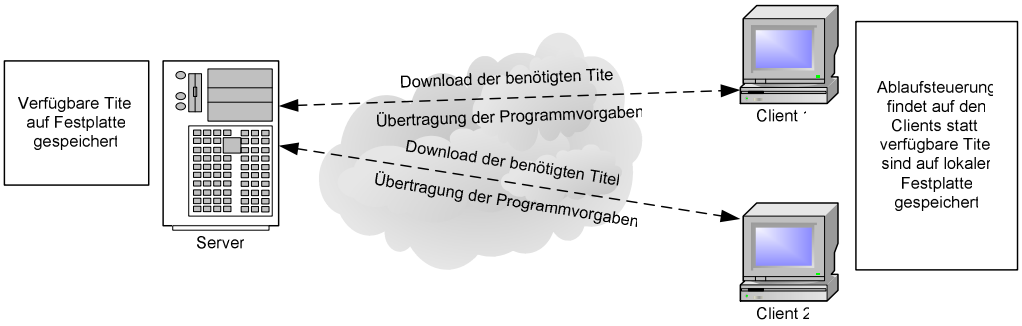

Abbildung 4.4b – DIRA Audiosystem

Man kann erkennen, dass dabei die Routinen zur Erstellung der Playlist auf die Clients ausgelagert werden. Diese übertragen die Vorgaben zur Gestaltung des Programms sowie die benötigten Titel vom Server auf die lokale Festplatte.

Durch die beschriebenen Modifikationen ergeben sich die im Folgenden beschriebenen Veränderungen gegenüber der TOX-Software:

4.4.1. Erhöhung der Zuverlässigkeit

Das Internet als Transportmedium ist zu unzuverlässig, um eine kontinuierliche,

störungs- und unterbrechungsfreie Beschallung zu garantieren (siehe Abs. 3.2). Der Client ist bei Nutzung des Streaming-Verfahrens darauf angewiesen, Daten mit einer durch die Kodierung des Audio-Materials festgelegten Bitrate *b* zu erhalten. Wird diese bei der Übertragung unterschritten, so kann dies kurzfristig durch die im Pufferspeicher *P* vorgehaltenen Daten abgefangen werden, längerfristig wird aber die Wiedergabe abbrechen.

$$Pufferzeit = \frac{P[byte]}{b[byte/s]}$$

Formel 4.4.1a – Größe des Streaming-Puffers

Eine Vergrößerung des Pufferspeichers löst dieses Problem nicht, da zum einen die Verzögerung ansteigt, zum anderen die so gewonnene Zeitspanne auch endlich ist.

Da die Ursachen für Fehler bei der Übertragung rund um den Globus auftreten können und sich somit völlig dem Einflussbereich von sowohl Server als auch Client entziehen, lässt sich eine garantierte Versorgung mit einer festen Bandbreite in der Praxis nicht erzielen.

Im Unterschied hierzu wird die Übertragung der Audiodaten beim DIRA-System durch Störung im Netzwerk kaum beeinträchtigt. Folgende Situationen sind dabei denkbar:

1. Verzögerungen oder Paketverluste: Auf Grund der Paketorientierung des Internets kann es vorkommen, dass Pakete nicht in der richtigen Reihenfolge oder verzögert ankommen. Auch der komplette Verlust einzelner Pakete ist möglich. Dies kann zum Beispiel geschehen, wenn Router ihre Routingtabellen aktualisieren. Beim Streaming von Audio- und/oder Videodaten können diese Verzögerungen durch den Einsatz von Puffern zu einem gewissen Grad abgefangen werden. Wird die Zeitspanne aber zu groß, so verwirft der Empfänger die verspäteten Pakete, da ein erneutes Anfordern zu lange dauern würde. Aus diesem Grund wird für Streaming meist das verbindungslose UDP-Protokoll eingesetzt, bei welchem der Sender keine

Empfangsbestätigung vom Empfänger erhält. Dies reduziert den Overhead bei der Übertragung.

Bei Einsatz des DIRA-Systems hingegen kann der Client die Daten mittels einer über TCP gesicherten Verbindung übertragen. Somit wird bereits auf Netzwerkebene sichergestellt, dass alle Pakete korrekt beim Empfänger ankommen. Solange der TCP-Stack in der Lage ist, ein verspätet eintreffendes Paket noch richtig in die Empfangswarteschleife einzuordnen, können selbst starke Verzögerungen toleriert werden. Auch so genannte Losses (verlorene Pakete) stellen kein Problem dar, da diese automatisch vom TCP-Stack des Clients erneut angefordert werden. Da die Wiedergabe im Gegensatz zum Streaming nicht zeitkritisch ist, stellen selbst schwere Beeinträchtigungen im Netzwerk kein Problem dar.

2. Überlastung: Ist eines der am Transport beteiligten Systeme überlastet, so sinkt die für jede Verbindung zur Verfügung stehende Bandbreite. Für das DIRA-System bedeutet dies, dass die Übertragung von neuen Titeln und der Meta-Daten länger dauert. Da aber die Wiedergabe der neuen Elemente erst nach der vollständigen Übertragung stattfindet, wirkt sich die erhöhte Transferzeit nicht auf das aktuell laufende Programm aus.

3. Ausfall: Wird die Verbindung zum Server während einer Übertragung unterbrochen, so würde im Falle einer Streaming-Übertragung die Wiedergabe abbrechen. Beim Einsatz der „Distributed Radio“ Technologie tritt bei der Wiedergabe keinerlei Beeinträchtigung auf, denn bei einem Übertragungsabbruch speichert der Client die unvollständige Datei (ohne sie jedoch in die Datenbank zu übernehmen) und versucht, diese beim nächsten Abgleich zu komplettieren.

4.4.2. Reduzierung der Datenmenge

Durch das Streaming von Audio und/oder Videodaten entstehen sehr große Datenmengen, welche ab einer gewissen Anzahl von gleichzeitigen Hörern nur schwer zu handhaben sind. Da beim DIRA-System kein kontinuierlicher Datenfluss von Server zu Client stattfindet, reduziert sich so die zu übertragende Datenmenge. Das Transfervolumen ist dabei abhängig von der Art des Programms. Vor allem Sender mit einer geringen Anzahl von Titeln in der Rotation eignen sich sehr gut für das DIRA-System. Der benötigte Traffik wird durch folgende Formel ausgedrückt:

$$Traffic = \sum_{\substack{Neue \\ Clients}} (Bitrate * Titel_{NeueClients}) + \sum_{Clients} (Bitrate * NeueTitel_{Clients})$$

Formel 4.4.2a – Benötigter Traffik beim DIRA-System

Die erste Summe bezeichnet die Initialübertragung, d.h. wenn ein neuer Client die für ihn vorgesehenen Titel anfänglich auf den lokalen Datenspeicher übertragen muss. Der zweite Teil beschreibt den für die Aktualisierung des Datenbestandes benötigten Datentransfer. Beim Vergleich dieser Formel mit Formel 3.2a wird deutlich, dass der Traffik beim DIRA-System unhabhängig von der Dauer der Wiedergabe ist. Aus diesem Grund kann man von der Gültigkeit von These 7 ausgehen.

Haben die Clients die unter Umständen recht umfangreiche erste Synchronisation abgeschlossen, so entsteht nur noch für neu eingespielte Titel weiterer Traffik

Anzumerken ist, dass in Formel 4.2.2a die Datenmenge für den Metadaten-Transfer nicht beinhaltet ist. Diese ist aber im Vergleich zur Nutzdatenmenge vernachlässigbar.

4.4.3. Individuelle Wiedergabe

Da beim DIRA-System die Logik für die Erstellung des Programms auf die Clients ausgelagert wurde, ergibt sich die Möglichkeit, für jeden angeschlossenen Client ein individuelles Programm zu erstellen. Dies bezieht sich hauptsächlich auf die Gestaltung der Playlist, kann aber auch individuelle Musikarchive beinhalten. Selbst ein komplett unterschiedlicher Bestand von Titeln pro Client würde gegenüber dem Streaming Einsparungen beim Tranfervolumen bringen, da sich wiederholende Stücke nicht erneut übertragen werden müssen.

Dies ist aber ein eher unrealistisches Einsatzszenario, im Normalfall wird man es mit einer Anzahl von Clients zu tun haben, welche über die gleichen Musikstücke verfügen und lediglich bei der Wiedergabe von etwa Werbeelementen variieren. Denkbar wäre in so einem Fall zum Beispiel regional angepasste Reklame. Wir werden in Abs. 5.6 auf solch ein Szenario eingehen.

4.4.4. Verzicht auf Live-Betrieb

Da kein kontinuierlicher Datenstrom zwischen Server und Client fließt, kann das Programm nicht in Echtzeit verändert werden. Das bedeutet, dass etwa ein neu eingespielter Programmbeitrag erst nach einem gewissen Zeitraum *t* von den angeschlossenen Clients gespielt werden kann. Dabei hängt *t* von der Häufigkeit der Synchronisation ab. In diesem Zusammenhang gilt zu beachten, dass die Latenz der Audio-Wiedergabe beim DIRA-System sogar sinkt. Da die Titel auf der lokalen Festplatte liegen, ist der Zeitraum zwischen Anweisung und tatsächlicher Wiedergabe wesentlich geringer (üblicherweise im Bereich einiger Millisekunden) als dies beim Streaming der Fall wäre – dort muss die Zeit für die Netzwerkübertragung dazugerechnet werden. Dieser Punkt sowie die in Abs. 4.4.1 beschriebenen Verbesserungen stellen eine Bestätigung für These 3 dar.

4.4.5. Zusammenfassung

Im Folgenden sollen die Vor- und Nachteile der beiden beschriebenen Systeme verglichen werden.

TOX-System

Vorteile	Nachteile
• Live-Betrieb • Clients können sehr einfach sein, z.B. Fraunhofer IP Kommandoanlage[98] • Einfaches Systemdesign • Unabhängig vom verwendeten System	• Hohe Traffikkosten • Störungsfreier Betrieb abhängig von Netzlast • Schlechte Skalierbarkeit • Suboptimaler Nutzungsgrad der Netzwerkressourcen

Tabelle 4.4.5a – Vorteile des TOX-Systems

Das verteilte Audiosystem DIRA wird durch folgende Punkte charakterisiert:

Vorteile	Nachteile
• Erheblich weniger Traffik • Unabhängigkeit von der Netzlast • Internetverbindung wird nur für den Abgleich benötigt • Hohe Ausfallssicherheit • Playlist kann individuell pro Client erstellt werden	• Kein Live-Betrieb • Komplexes Systemdesign • Lokale Datenspeicher benötigt • Benötigte Titel müssen anfangs erst übertragen werden

Tabelle 4.4.5b – Vorteile des DIRA-Systems

[98] http://www.fokus.gmd.de/research/cc/cats/products/Flyer-Kommandoanlage.pdf

Vereinfacht kann man sagen, dass für den Betrieb eines solchen Systems die Anforderungen an die Hard- und Software der Clients steigen. Da die Aufgaben wie das Erstellen einer Playlist nun nicht mehr auf dem Server ausgeführt werden, spricht man hier von verteiltem Rechnen oder "Distributed Computing". Da das Einsatzfeld des DIRA-Systems im Bereich der Klangerzeugung durch die Clients liegt, kann man in diesem Fall analog von einem „Distributed Radio Network“ sprechen. Im Folgenden sollen nun die benötigten Hard- und Softwarekomponenten beschrieben werden.

4.5. Implementierung

4.5.1. Verwendete Hardware

Für den Server kam, anders als bei der Magisterarbeit, ein gekauftes Gerät zum Einsatz, welches in einem Rechenzentrum in Berlin untergebracht wurde (Colocation).

Der Rechner ist folgendermaßen ausgestattet:

- 1HE Gehäuse mit 250W Netzteil
- MSI 9128 Mainboard, Socket 478
- Infineon 256 MB DDR-Speicher
- Intel Celeron 1,8 GHz Prozessor
- 2 * Maxtor 6Y060L0 Laufwerke mit 60 GB Kapazität, 7200 U/min

Die beiden Festplatten sind als RAID-1 Verbund konfiguriert, so dass bei Ausfall einer Platte die Funktionsfähigkeit des Gesamtsystems nicht beeinträchtigt wird. Wenn allerdings (wie im Oktober 2003 geschehen) beide Platten an einem Tag ausfallen, so werden auch bei dieser Konfiguration die Grenzen der Hardware aufgezeigt. Durch das 1HE Gehäuse ist das Gerät gerade einmal 2,5 cm hoch und

kann so sehr kostengünstig untergestellt werden.

Abbildung 4.5.1a - Bild des Servers

Es sei angemerkt, dass die Rechenleistung der Hardwareausstattung in etwa die niedrigste ist, die man bei neuen Systemen noch kaufen kann. Da aber der Großteil der Arbeitsschritte auf die Clients ausgelagert wird, ist sie bei weitem ausreichend.

Die Anbindung erfolgt über eine 100 MBit/s Fast Ethernet Karte. Da es sich beim Provider vor Ort (Level-3[99]) um eine amerikanische Firma handelt, die in Europa nur ein eingeschränktes, eigenes Netzwerk besitzt, ist die Geschwindigkeit der Anbindung suboptimal.

Der Client ist folgendermaßen ausgestattet:

- Standart-PC Gehäuse mit 250W Netzteil
- Asrock K7VM2 Mainboard
- 128 MB No-Name SDR-Speicher
- Athlon XP 1700+ Prozessor
- Samsung SP0411N Festplatte mit 40 GB Kapazität, 7200 U/min
- AVM Fritz PCI ISDN-Karte

[99] http://www.level3.com/2577.html

Auch dieses Gerät ist technisch am unteren Ende der Leistungsklassen, für das DIRA-System aber ausreichend. Das Board verfügt über eine integrierte Netzwerkschnittstelle, die etwa bei vorhandenem DSL-Anschluss statt der ISDN-Karte benutzt werden kann. Weiterhin ist es mit einer integrierten Soundkarte ausgestattet, über welche das Audiosignal wiedergegeben wird.

4.6. Verwendete Software

Bei der Auswahl der verwendeten Software wurde großer Wert auf deren freie Verfügbarkeit gelegt. Das bedeutet, dass alle benutzten Programme unter der "General Public Licence" (GPL) oder einer vergleichbaren Lizenz verfügbar sind. Vereinfacht ausgedrückt bedeutet das, die Programme liegen im Quellcode vor und es wird anderen Programmierern ausdrücklich erlaubt, diesen für eigene Projekte zu benutzen - mit der einzigen Einschränkung, dass dieses neue Projekt ebenfalls unter der GPL veröffentlicht wird. Aus diesem Grund werde ich sämtliche Quelltexte nach Veröffentlichung dieser Arbeit auf meiner Webseite zugänglich machen.

Als Betriebssystem kommt sowohl auf dem Server als auch den Clients Debian[100] zum Einsatz. Diese Linux-Distribution ist das Ergebnis einer globalen Zusammenarbeit zwischen Entwicklern. Sie unterscheidet sich in folgenden Punkten von anderen Distributionen:

- Es kommt ausschließlich freie Software zum Einsatz. Distributionen wie RedHat[101] oder Suse[102] setzen häufig proprietäre Software für die Installation und Verwaltung des Systems ein. Diese ist meist nicht im Quelltext vorhanden und darf auch nicht frei kopiert werden.

[100] http://www.debian.org

[101] http://www.redhat.de

[102] http://www.suse.de

- Der Paketmanager *apt-get* erleichtert die Installation von Softwarepaketen enorm, indem er zusätzlich benötigte Programmodule (Dependencies) automatisch erkennt und installiert.
- Die Installation und das Aktualisieren von Programmen lassen sich sehr bequem über das Internet bewerkstelligen. So bewirken etwa die Befehle „apt-get update“ und „apt-get upgrade“ eine automatische Aktualisierung der installierten Softwarepakete.
- Alle verfügbaren Programmpakete sind in drei Zweige stable, testing und unstable eingeteilt. Wird eine neue Programmversion in die Distribution aufgenommen, so wird sie zuerst im unstable Bereich geführt. Hat sie sich dort einige Zeit bewährt, wird sie nach testing verschoben und nach relativ langer Testphase letztendlich in stable aufgenommen. Das hat zur Folge, dass die Versionsnummern der in stable geführten Programme teilweise weit hinter der gerade aktuellen Version zurückliegen - dafür kann man auch davon ausgehen, dass sie extrem stabil laufen. Der Server läuft im Moment seit 267 Tagen, ohne dass bisher ein Programm abgestürzt wäre[103].

Folgende Software kommt dabei zum Einsatz:

4.6.1. Apache

Da die Administration des DIRA-Systems komfortabel über einen Webbrowser erfolgen soll, ist es erforderlich, einen Webserver zu betreiben, der die entsprechenden Seiten zur Verfügung stellt. Dabei führt unter Linux kein Weg an dem Programm *Apache* vorbei.

Dieser Webserver zählt zu den am häufigsten eingesetzten Linux Programmen überhaupt. Er gilt als sehr sicher und leistungsfähig und bietet eine Vielzahl von Schnittstellen für weitere Software. Im November 2003 betrug der Marktanteil global 67,41 Prozent, der Webserver von Microsoft (*Internet Information Server*)

[103] [KOM1998], S. 447 - 765

kam auf lediglich 21,02 Prozent[104]. Für eine Software, die kostenlos ist und letztendlich niemandem gehört, eine herausragende Leistung.

Wegen des Debian Grundsatzes, alle Programme erst ausführlich zu testen, kommt die Version 1.3.26 zum Einsatz. Diese benutzt anders als die aktuelle Version 2 noch das alte Multi-Process Verfahren. Dabei startet der sogenannte Master-Prozess eine frei definierbare Anzahl von Unterprozessen mittels des fork() Befehls. Diese Unterprozesse sind für die Beantwortung der Web-Anfragen zuständig. Mittels einer Scoreboard genannten Datei kann der Master-Prozess feststellen, wie beschäftigt die einzelnen Prozesse sind und bei Bedarf (also vielen Anfragen) weitere starten.

Beim von *Apache 2* eingesetzten Multithread-Verfahren startet der Master-Prozess je nach Bedarf unterschiedlich viele Threads, welche sich aber, anders als beim Unterprozess-Verfahren, die Ressourcen, den Code sowie den Arbeitsspeicher des Masters teilen. Daher benötigen sie wesentlich weniger Speicher als die vergleichbare Anzahl von Prozessen. Für die Entwicklung des Systems ist diese Schwachstelle akzeptabel, für den Praxiseinsatz wäre ein Umstieg auf die neuere Version erwägenswert.

Die prinzipielle Funktionsweise eines Webserver soll in folgender Grafik dargestellt werden:

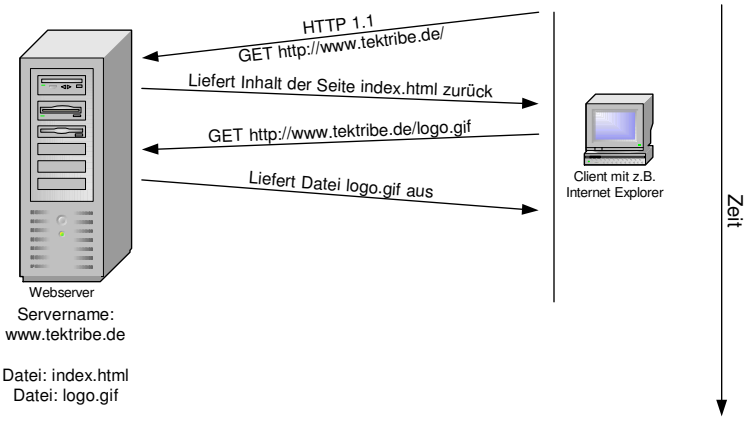

Abbildung 4.6.1a - Schematische Darstellung der Kommunikation zwischen Webserver und Client

[104] http://news.netcraft.com/archives/web_server_survey.html

Der Webserver wartet standardmäßig auf Port 80 des TCP-Protokolls auf eingehende Anfragen. Fordert nun ein Benutzer mittels seines Webbrowsers eine Seite an, so liefert der Server diese als HTML-formatierten Text aus. Der Browser des Benutzers wertet (parsed) diesen Text aus und stellt fest, dass er für die vollständige Darstellung noch die Datei logo.gif benötigt, welche er vom Server anfordert. Daraufhin ist in diesem einfachen Beispiel der Aufbau der Seite abgeschlossen. Durch das Klicken auf einen Link kann der Client nun zum Beispiel weitere Dateien anfordern.

Der Nachteil an diesem Verfahren ist, dass sich so nur statische Seiten anzeigen lassen. Will man etwa den Inhalt einer aufzubauenden Seite abhängig vom Suchbegriff des Anwenders machen (z.B. Suchmaschine), so ist es erforderlich, diese individuell entsprechend des Begriffes zu erstellen.

4.6.2. PHP

Benutzte man früher für solche Aufgaben noch häufig das so genannte "Common Gateway Interface", kurz CGI, um etwa Eingaben des Benutzers an ausführbare Programme weiterzugeben, so ist man heute dazu übergegangen, die Programmiersprache gleich in den Webserver zu integrieren. Dies verbessert die Performance des Systems und ist zudem wesentlich weniger anfällig für Fehler.

Die wichtigsten Sprachen in diesem Bereich sind Microsofts *ASP* (Active Server Pages), Suns *JSP* (Java Server Pages) und die frei verfügbare Sprache *PHP* (Personal Hypertext Processor). *PHP* ist für den *Apache* Server als ladbares Modul verfügbar, wodurch man den Webserver einfach um die Möglichkeiten von *PHP* erweitern kann. Die eingesetzte Version 4.1.2 entspricht dem letzten Debian stable Release.

PHP ist ein sehr mächtiges Werkzeug zum Gestalten von Webseiten, für das DIRA-System werden wir die Sprache für folgende Funktionen benutzen:

- Datenbankoperationen: *PHP* ermöglicht es, vor dem Ausliefern einer Webseite in einer Datenbank Informationen abzurufen und diese in die Seite zu integrieren. Folgendes Beispiel sucht nach einem Künstler oder einem Titel in einer *MySQL*-Datenbank:

```
<?php
$result = mysql_query (SELECT * FROM archive WHERE
artist LIKE '$entry' or songtitle LIKE '$entry'");
?>
```

Hier wird die Variable $result mit dem Ergebnis der Suche nach dem Künstler oder eines Titels geladen. Die <?php und ?> Tags dienen dazu, dem Webserver zu zeigen, was eine Programmanweisung und was normaler HTML-Quelltext ist.

- Benutzerauthentifizierung: Um die verschiedenen Benutzer des DIRA-Systems identifizieren zu können, wird mit Hilfe von *PHP* ein so genanntes Cookie im Browser des Anwenders gespeichert. Über dieses sind eine eindeutige Identifikation sowie die Vergabe von unterschiedlichen Zugriffsrechten möglich.

- Dateioperationen: Um etwa neue Dateien umzukopieren oder in ein anderes Format zu überführen, macht das DIRA-System Gebrauch von den Befehlen des Linux Betriebssystems. Ein Beispiel dafür wäre folgende Anweisung zum Neukodieren eines hinzugefügten Titels:

```
system("perl    /usr/bin/mp3check.pl    --reencode-
bitrate=128 --reencode-freq=44100 --ignore-spaces -
-ignore-track-nums $myfile 2>&1");
```

Der system() Befehl dient dazu, Systemroutinen aufzurufen – in diesem Fall das Programm *mp3check.pl*, welches eine MP3-Datei überprüft und neu

kodiert.

Auf die verwendeten *PHP*-Befehle werden wir in Abschnitt 4.8.1 noch näher eingehen. *PHP* und *Apache* werden nur auf dem Server benötigt.

4.6.3. MySQL

Um die für das System benötigten Daten (z.B. Titel eines Musikstücks) zu speichern und auszulesen, bietet sich die Benutzung einer Datenbank an. Die einfachste Lösung wäre dabei, alle Informationen in einer Textdatei zu speichern, man würde in diesem Fall von einer sequentiellen Datenbank sprechen. Wesentlich komfortabler und übersichtlicher ist die Aufnahme in eine relationale Datenbank, da hier die Daten zum einen strukturierter gespeichert werden, zum anderen, wie der Name schon sagt, zueinander in Verbindung stehen[105].

Datenbanken bilden die Datensätze (ähnlich wie bei einer Tabellenkalkulation) immer in Form von Tabellen ab. Relational wird eine Datenbank dadurch, dass sie nicht nur aus einer großen Liste sondern aus mehreren kleinen Tabellen besteht. Die Datenmengen können dadurch gering gehalten werden, da die Daten nicht doppelt geführt werden müssen. Mit so genannten Schlüsselfeldern ("keys") werden zwischen den Tabellen Verknüpfungen erstellt. Dies wird in folgender Grafik deutlich:

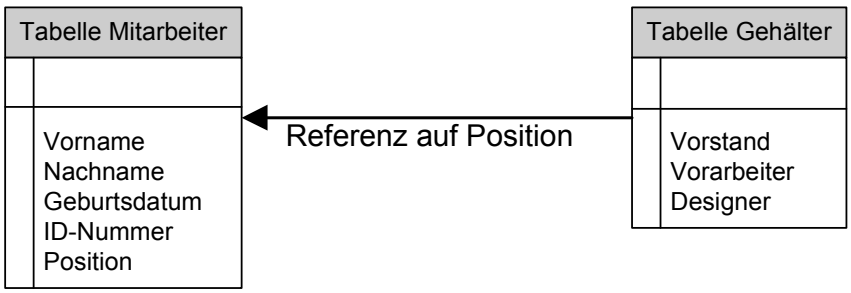

Abbildung 4.6.3a - Schema einer MySQL Datenbank

[105] http://sql.idv.edu/thema/dbgrundlagen/begriffsklaerung.htm

In diesem sehr einfachen Beispiel würde man aus der Position eines Mitarbeiters in der Tabelle "Mitarbeiter" dessen Gehalt aus der Tabelle "Gehälter" bestimmen können. Man spricht daher von einer Relation des Schlüssels "Position" auf einen Einträg einer anderen Tabelle[106].

Damit man leicht auf die gespeicherten Daten zugreifen kann, wurde von der amerikanische Standardisierungsbehörde ANSI die Sprache SQL entwickelt. Diese ermöglicht es, mittels einfacher Textbefehle Informationen in eine Datenbank einzutragen oder auszulesen. Ein Beispiel hierfür wäre etwa:

```
INSERT INTO mitarbeiter
(vorname,nachname,geburtstag) VALUES
(`Frank`,`Mueller`,`19.10.1960`);
```

Damit würde man einen neuen Mitarbeiter in die Datenbank aufnehmen, die Werte für ID-Nummer und Position blieben leer. *MySQL* beherrscht nicht den kompletten Funktionsumfang der ANSI-Vorgaben, einige komplexe Operationen wie etwa Transaktionen, Stored Procedures oder Trigger werden (zumindest in dieser Version) noch nicht unterstützt. Daher verwenden Firmen mit sehr großen Datenbeständen (etwa eBay[107] oder Amazon[108]) kommerzielle Datenbanken wie Oracle oder db2. Diese bieten wesentlich mehr Funktionen, sind dafür aber auch sehr teuer. Für diese Arbeit hat sich *MySQL* als absolut ausreichend erwiesen[109].

Die beim DIRA-System eingesetzte Versionsnummer beträgt 3.23.49 und entspricht somit der Vorgabe aus dem stable-Zweig. Der Zugriff von Seiten der *PHP*-Skripten wird durch das php4-mysql *Apache*-Modul ermöglicht, welches Datenbankanfragen zur Laufzeit eines Skriptes ermöglicht. Die Versionsnummer lautet hier 4.1.2. *MySQL* kommt sowohl auf dem Server als auch auf den Clients zum Einsatz.

---

[106] [BAS2000], S. 18ff

[107] http://www.ebay.de

[108] http://www.amazon.de

[109] [WIG1977], S. 349ff

4.6.4. PureFTPd

Der *PureFTP* Dämon[110] ist so genanntes "Lightweight" FTP-Serverprogramm. Es stellt eine Weiterentwicklung der TrollFTP-Software dar und bietet eine Schnittstelle, über welche die Clients neue Musikstücke oder sonstige Programmelemente auf die lokale Festplatte transferieren können. Der Download findet dabei über das File Transfer Protocol (FTP) statt.

Die Entscheidung für *PureFTP* (es gibt über 20 FTP-Serverprogramme für Linux) fiel aus folgenden Gründen:

- Hohe Sicherheit: Bisher wurde noch kein *PureFTP*-System kompromittiert - im Gegensatz etwa zum Server *wu-ftpd* der Washington University[111], bei welchem eine Reihe von Fehlern entdeckt wurden.
- Geringe Komplexität: Das "Lightweight" deutet darauf hin, dass der Funktionsumfang dieser Software geringer ist als der anderer vergleichbarer Programme. Da für das DIRA-System nur die grundlegenden Funktionen des FTP-Protokolls benötigt werden (Login, Download von Dateien) stellen weitergehende Optionen nur eine unnötige Komplizierung dar.
- Leichte Installation und Konfiguration: Anders als etwa bei *ProFTP*[112] oder *wu-FTP*, die umfangreiche Konfigurationsarbeit benötigen (Anlegen von Verzeichnissen mit den vom Benutzer benötigten Programm etc.), lässt sich *PureFTP* mit nur wenigen Änderungen der Konfigurationsdatei in Betrieb nehmen.
- Chroot-Umgebung: Die Software beherrscht das Umsetzen des Wurzelverzeichnisses abhängig vom Benutzer. Das bedeutet, dass der Client

[110] http://www.pureftpd.org/

[111] http://www.wu-ftpd.org/

[112] http://www.proftpd.net/

nicht aus seinem Heimatverzeichnis "ausbrechen" und eventuell andere Dateien lesen kann.

4.6.5. Perl

Auf Grund der sehr guten Erfahrungen mit der "Practical Extraction And Reporting Language" bei der Programmierung der Software zu meiner Magisterarbeit wird diese auch ausgiebig im DIRA-System eingesetzt. *Perl* wurde bereits 1987 vom Linguisten Larry Wall entwickelt und hat sich seitdem zur wohl am häufigsten eingesetzten Skriptsprache im Unix-Bereich durchgesetzt. Das hat eine Reihe von Gründen:

- Schnelle Programmentwicklung, da nur eine sehr laxe Variablen- und Typenüberprüfung stattfindet.
- Viele Strukturen orientieren sich stark an gesprochener Sprache. Ein Beispiel wäre etwa `Befehl UNLESS Bedingung.` Die Anweisung "Führe die Tätigkeit aus, solange bis Bedingung eintritt" klingt wesentlich sinnvoller (zumindest für Menschen) als "Falls Bedingung nicht eintritt, führe die Tätigkeit weiter aus" (if ... then Schleife).
- Unter der Adresse www.cpan.org[113] finden sich für quasi alle denkbaren Anwendungszwecke zusätzliche Module, die *Perl* um verschiedene Funktionen erweitern.
- *Perl*-Programme sind sehr leicht portabel, das heißt, man kann sie leicht auf andere Systeme übertragen.
- Da *Perl* in der Linux-Welt schon lange genutzt wird, findet man umfangreiche Dokumentation und Programmbeispiele, was das Erlernen vereinfacht.

[113] http://www.cpan.org

Kritiker der Sprache führen an, dass *Perl* (auch auf Grund der fehlenden Objektorientierung) zu einem unsauberen Programmierstil führt und dass *Perl*-Programme schlecht lesbar sind. Ganz Unrecht haben sie dabei wohl nicht, Programmanweisungen wie

```
s/^([^ ]*) *([^ ]*)/$2 $1/
```

erschließen sich nicht auf Anhieb (dieser reguläre Ausdruck vertauscht die beiden ersten Wörter einer Zeile). Allerdings ist diese Anweisung an Kompaktheit wohl kaum zu übertreffen.

Wir werden im Rahmen des DIRA Programms nur wenig Gebrauch von regulären Ausdrücken machen, daher ist der Quelltext leicht verständlich. Die eingesetzte Programmversion von *Perl* beträgt 5.6.1, zusätzlich wurde das DBI Modul in der Version 1.2.1 installiert. Dieses ermöglicht es, aus *Perl*-Skripten auf eine *MySQL* Datenbank zuzugreifen. Dieser Zugriff gestaltet sich ähnlich der *PHP*-Programmierung und wird in Abs. 4.11.1 weiter behandelt.

Perl wird hauptsächlich auf dem Client eingesetzt, dort dient die Sprache der Steuerung der Wiedergabe sowie dem Abgleich der Datenbestände mit dem Server. Auf dem Server beschränkt sich der Einsatz auf die Überprüfung und das Rekodieren der eingespielten Titel.

4.6.6. Jukepeg

Da im Rahmen der *Perl* und *PHP* Programmierung auch auf bereits vorhandene Erweiterungen für die jeweilige Sprache zurückgegriffen wird, liegt es nahe, dasselbe für die Software zum Abspielen der MP3-Dateien zu tun. Die Anforderungen an diese sind folgende:

- Steuerung der Wiedergabelisten über Skripten

- Blenden zwischen den Titeln
- Bedienbarkeit ohne grafische Oberfläche

Nach einiger Suche bin ich auf das Programm *Jukepeg*[114] der Firma BitBox gestoßen. Es erfüllt alle genannten Anforderungen und würde sogar den Betrieb mit mehreren Soundkarten unterstützen. Die Steuerung erfolgt über ein so genanntes Socket, einer Schnittstelle zur Kommunikation zwischen verschiedenen Programmen. Ist das Programm aktiv (es läuft als Dämon im Hintergrund), so legt es eine Schnittstellen-Datei an (das Socket), in welche sowohl Befehle an das Programm geschrieben werden als auch umgekehrt die Ausgaben des Programms erfolgen. Die Wiedergaberoutinen für die MP3-Dateien sind dem *FreeAmp*-Player[115] entnommen, was auf Grund der GPL-Lizenz für beide Programme kein Problem darstellt.

Jukepeg läuft auf dem Client-Rechner und dient der Wiedergabe der lokal gespeicherten Titel. Auf die genaue Interaktion mit dem System wird in Abs. 4.12 noch genauer eingegangen. Kurz zusammengefasst gibt es eine Reihe von Befehle, die den Dämon steuern:

- queue(Dateiname): Damit wird die mit Dateiname bezeichnete Datei in die interne Warteschlange übernommen.
- overlap(Zeit): Gibt an, wie lange der Übergang zwischen 2 Titeln andauert
- next: Beendet die Wiedergabe des laufenden Titels und springt zum nächsten Eintrag in der Warteschlange
- flush: Löscht alle Einträge in der Warteschlange

Die eingesetzte Versionsnummer beträgt 1.9. Das Programm wird zwar nicht mehr weiter entwickelt, es verrichtet seinen Betrieb allerdings einwandfrei. Daher besteht auch kein Grund, auf eine andere Software auszuweichen. Es ist lediglich zu

[114] http://www.bitbox.co.uk/jukepeg/

[115] http://www.freeamp.org

beachten, dass die Software keine MP3-Dateien abspielen kann, welche mit variabler Bitrate (VBR) kodiert worden sind - der Programmierer begründet dies mit Problemen beim Überblenden von Titeln. Daher ist bereits serverseitig darauf zu achten, dass solche Titel nicht in das System aufgenommen werden.

4.6.7. Lame und mp3check.pl

Zu diesem Zweck kommen die beiden Programme *Lame* und mp3check.pl zum Einsatz. Sie sind für die Sicherstellung der richtigen Bitrate sowie die Fehlerfreiheit der benutzten MP3-Dateien zuständig. Bei *Lame*[116] handelt es sich um eine Open Source Implementierung eines MP3-Codecs. Der Name steht für „Lame ain't an MP3 Encoder", worin sich zum einen die bei Informatikern beliebte Rekursion verbirgt, zum anderen die Tatsache, dass das Projekt als Patch für die vorhandenen ISO-Vorgaben für das MP3-Format begann. Diese sind aber mittlerweile nicht mehr notwendig, *Lame* ist ein vollwertiger MP3-Encoder, der sogar schneller als der Referenz-Codec der Fraunhofer Gesellschaft arbeitet[117] .

Mp3check.pl[118] ist ein kleines *Perl*-Skript von Ryan C. Gordon, das unter der GPL lizenziert ist. Dieses Programm erlaubt es, MP3 Dateien auf formale Fehler zu untersuchen und sie in eine neue Bitrate zu konvertieren.

Der Aufruf von zum Beispiel

```
./mp3check.pl --reencode-bitrate=128 --reencode-freq=44100 titel.mp3
```

würde die Datei title.mp3 mit Hilfe von *Lame* neu kodieren, und zwar mit einer Bitrate von 128 KBit/s und einer Sample-Auflösung von 44100 Hz. Leider überprüft das Programm dabei nur, ob Länge, Bitrate usw. mit den im Header angegebenen

[116] http://www.mp3dev.org/mp3/

[117] http://www.modatic.net/audio/mp3_encoder_comparison.php

[118] http://www.icculus.org/mp3check/mp3check

Werten übereinstimmt, nicht allerdings, ob jeder einzelne Frame auch fehlerfrei dekodiert werden kann.

In der derzeitigen Programmversion werden daher noch alle hochgeladenen Titel neu kodiert. Für ein Produktivsystem ist dies weniger wünschenswert, da der Server durch das Kodieren stark belastet wird. Sinnvoller wäre es, die Dateien erst auf Fehler und richtige Bitrate zu überprüfen. Sind sie beschädigt, sollten sie verworfen werden, nur bei einer falschen Bitrate wird eine Neu-Kodierung durchgeführt.

4.6.8. mp3_check

Dieses sehr ähnlich lautende Programm (man beachte den Unterstrich) überprüft, ob die im Header einer MP3-Datei angegebenen Informationen gültig sind und gibt diese aus. Sowohl das *mp3check.pl* Skript, also auch die Uploadroutinen der Webseite machen davon Gebrauch.

4.6.9. MP3Gain

Bei der Wiedergabe des Programms wirkt sich die Tatsache, dass die Lautstärke verschiedener Titel stark variieren kann, negativ auf den Gesamthöreindruck aus. Weiterhin ist es im Radiobereich üblich, Werbe- und Programmelemente lauter als die Musik auszuspielen, um diese mehr in den Vordergrund zu rücken[119].

Aus diesem Grund habe ich auf das Programm *MP3Gain* des Autors Glen Sawyer[120] zurückgegriffen. Es erlaubt das Normalisieren von MP3-Dateien, ohne diese dafür dekodieren zu müssen. Dabei wird nicht nur das lauteste Sample gesucht (Peak-

[119] [MCR1994], S. 213

[120] http://mp3gain.sourceforge.net/

Normalisierung), sondern das Programm errechnet die durchschnittlich wahrgenommene Lautstärke. Nach dem Upload eines Titels wird dieser mittels

```
mp3gain -r -q -c  "$myfile"
```

angepasst. Die Schalter stehen für automatische Anpassung (-r), das Ignorieren von Warnmeldungen (-c) sowie den Quiet-Modus (-q). Im Praxiseinsatz erwies sich die Funktionalität der Software als sehr zuverlässig. Eleganter wäre natürlich der Einsatz einer Kompressor-Software, welche die Lautstärke dynamisch bei der Wiedergabe regelt. Da allerdings in *Jukepeg* keine Schnittstelle dafür zur Verfügung stand und somit große Veränderungen im Quelltext nötig gewesen wären, habe ich mich für die einfachere Lösung entschieden. Interessant in diesem Zusammenhang ist noch, dass sich das eigentlich für Windows entwickelte *MP3Gain* problemlos unter Linux übersetzen ließ.

4.6.10. ntp-Server

Da die Synchronisation der Clients über sogenannte Timestamps (mehr dazu in Abs. 4.11.3) gesteuert wird, ist es wichtig, dass sowohl Server als auch Clients die richtige Uhrzeit verwenden. Auf dem Server kommt dazu das *ntp*-Paket zum Einsatz. Es besteht aus dem Basispaket *ntp* (Version 4.1.0-8) sowie dem *ntp-simple* (ebenfalls 4.1.0-8) Serverprogramm. Die Clients nutzen nur das *ntpdate* Programm aus dem *ntp* Paket.

Dieses Serverprogramm läuft dabei permanent im Hintergrund und überprüft, ob die Systemuhr richtig geht. Dazu benutzt es über das Internet zugängliche Zeitserver, welche mit einer Atomuhr gekoppelt sind. Beim DIRA-System werden die beiden Server der physikalisch-technischen Bundesanstalt in Braunschweig[121] (ntp1.ptb.de bzw. ntp2.ptb.de) verwendet.

[121] http://www.ptb.de/de/org/q/q4/q42/ntp/ntp_main.htm

4.7. Zusammenwirken der Komponenten

Bevor wir nun auf die genaue Funktionsweise der verwendeten Software und der Skripten eingehen, soll zuerst der grobe Zusammenhang zwischen den verschiedenen Modulen beschrieben werden. Dabei gilt es die Server- von den Client-Komponenten zu unterscheiden. Zusätzlich soll noch die Kommunikation zwischen beiden Instanzen umrissen werden.

Wie bereits in Abs. 4.2 beschrieben, besteht der Ansatz beim DIRA System darin, einen Server mit einer statischen IP-Adresse und schneller Internetanbindung sowie eine Vielzahl von minimal ausgestatteten Clients mit nicht-permanenter Internetanbindung zu nutzen.

4.8. Komponenten des Servers

Der über das Internet erreichbare Server stellt sozusagen die Koordinierungsstelle für die Wiedergabe der Programmelemente seitens der Clients dar. Weiterhin dient er als Schnittstelle für den Anwender, da über die vorhandenen Webseiten eine Interaktion möglich ist. Alle weiteren Funktionen laufen für den Anwender transparent ab. Die einzelnen Komponenten greifen wie in Abb. 4.8a dargestellt ineinander.

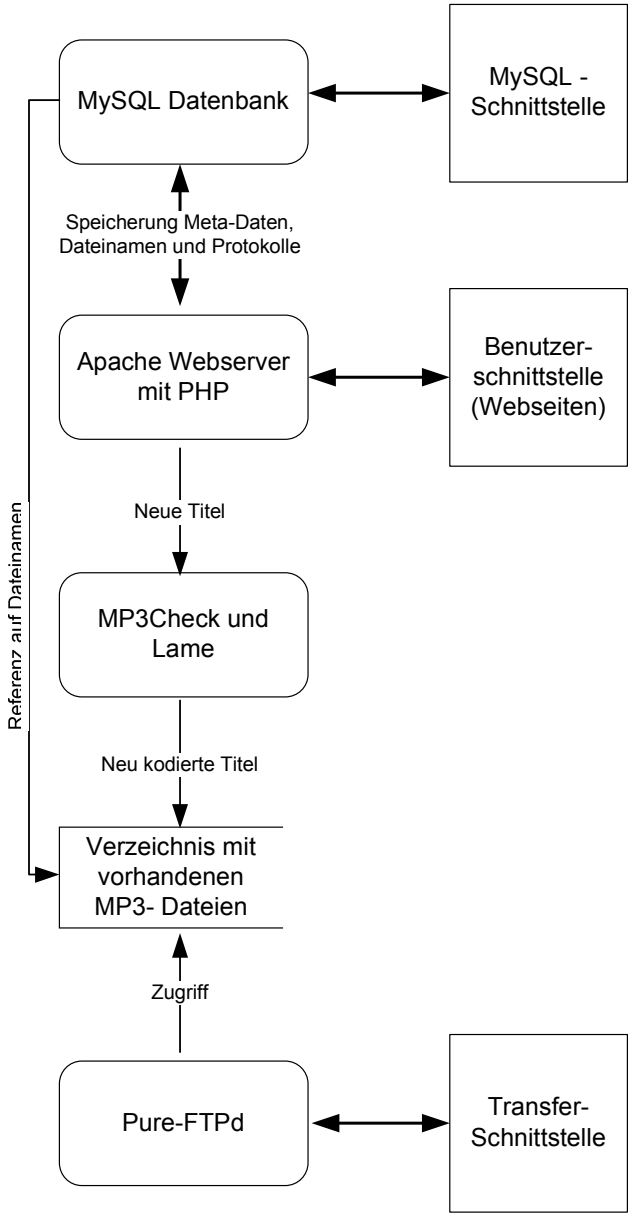

Abbildung 4.8a – Schematische Darstellung der Serverkomponenten

Hier wird deutlich, dass der Server über drei verschiedene Schnittstellen verfügt. Die beiden internen Schnittstellen (Transfer- bzw. *MySQL*-Schnittstelle) dienen der Kommunikation mit den Clients, die externe (entsprechend dem Benutzerinterface) ermöglicht die Bedienung des Systems.

4.8.1. Ausgewählte PHP-Skripten

Die Webseiten dienen dazu, das System für den Benutzer auf einfache Weise zugänglich zu machen. Wird etwa die Abfolge der Titel geändert, so wird diese Veränderung in der *MySQL*-Datenbank eingetragen. Die weiteren Programmmodule greifen ebenfalls auf die Datenbank zu und erfahren so die neue Reihenfolge der Titel.

Man kann daher sagen, dass die Datenbank als Container für alle anfallenden Informationen dient - mit Ausnahme der vorhandenen MP3-Dateien, diese werden lediglich als Datei in einem speziell dafür vorgesehenen Ordner gespeichert. Die Funktion jeder einzelnen Webseite zu erklären, würde den Umfang dieser Arbeit allerdings sprengen, daher soll nur auf einige häufig verwendete Programmabschnitte und Besonderheiten eingegangen werden.

4.8.2. Authentifizierung

Da das System nicht für jedermann zugänglich sein soll, ist es notwendig, die Berechtigung eines Benutzers zu überprüfen. Die dazu erforderlichen Schritte **A**uthentication, **A**uthorisation und **A**ccounting werden häufig zur Abkürzung AAA zusammengefasst.

Die einfachste Form der Authentication wäre der Schutz per .htaccess bzw. .htpasswd Datei. Diese beiden Dateien müssen dabei in das zu schützende Verzeichnis des Servers kopiert werden, der *Apache* Webserver verlangt daraufhin einen Benutzernamen und ein Passwort, bevor er die Seite anzeigt[122]. In diesem Fall würde die Authorisation durch den Webserver erfolgen.

Diese Art der Passwortabfrage wird als "basic authentication" bezeichnet und ist für das DIRA-System aus folgenden Punkten nicht ausreichend:

[122] http://selfhtml.teamone.de/diverses/htaccess.htm

- Mehrfachlogin möglich: Es ist möglich, dass mehrere Benutzer mit der gleichen Kennung simultan auf das System zugreifen. Dies ist unerwünscht, da dadurch Komplikationen etwa bei der Programmgestaltung auftreten können.
- Keine Abstufung der Benutzerrechte: Es ist wünschenswert, die Zugriffsrechte feiner abstufen zu können. So soll etwa der Benutzer Redakteur weniger Veränderungen am Programm durchführen können als etwa der Programmchef.
- Kein automatischer Logout: Einmal angemeldet, bleibt eine Sitzung mittels .htaccess Authentifizierung bis zum Schließen des Webbrowser gültig. Auf einem Rechner mit vielen verschiedenen Benutzern (z.B. Internetcafé) kann dies zu Sicherheitsproblemen führen, da spätere Benutzer mittels des "Zurück" Buttons ohne Passwortabfrage auf die Seite zurückkehren können.
- Kein Tracking: Für ein größeres System ist es wichtig, die Benutzerbewegungen zwischen den Seiten nachzuvollziehen - etwa um die Bedienbarkeit zu verbessern. Dies ist mittels .htaccess nicht möglich.
- Kein Accounting: Für ein kommerzielles System bietet diese Art der Anmeldung unzureichende Möglichkeiten des Accountings. Das bedeutet, dass Aktionen eines einzelnen Users schwer abzurechnen sind.

Aus diesen Gründen habe ich mich für eine Cookie-basierte Authentifizierung entschieden. Dabei wird bei erfolgreicher Anmeldung an das System auf dem Rechner des Anwenders ein so genanntes Cookie mit einer 32-stelligen Zufallszahl gespeichert. Der Server speichert diese Zahl ebenfalls in der *MySQL*-Datenbank. Bevor nun eine Seite angezeigt wird, überprüft der Server das Cookie des Clients mit dem lokal gespeicherten - stimmen beide überein, wird die Seite angezeigt. Im Quelltext sieht das folgendermaßen aus:

```
/* Cookie und SessionID */
if(!$sessionid)
{
mt_srand((double)microtime()*1000000);
$sessionid = md5(str_replace(".","",$remote_addr) +
mt_rand(100000,999999));
setcookie("sessionid", $sessionid, time()+600);}
```

Falls das Cookie $sessionid nicht vorhanden ist, wird eine Zufallszahl erzeugt, welche zusätzlich zur IP-Adresse des Benutzer addiert wird. Von dieser Zahl wird durch den md5()-Befehl ein 32-stelliger MD5-Hash erzeugt und mittels setcookie() gespeichert. Die time()+600 gibt das Verfallsdatum des Cookies an. In diesem Fall würde es nach 600 Sekunden ungültig werden und der Benutzer müsste sich neu anmelden.

```
$mysql_link = mysql_connect("localhost","user","pw") or
die ("Failed to connect to host!");

mysql_select_db("dira", $mysql_link) or die ("Failed to
connect to database!");

$result = mysql_query("SELECT * FROM adminlog WHERE
userid = '$sessionid'");
```

Hier wird in der Datenbank überprüft, ob die vorhandene Sessionid bereits gespeichert ist. Ist dies nicht der Fall, so wird die Abarbeitung des Skriptes gestoppt und eine Fehlermeldung ausgegeben:

```
        if (!$row = mysql_fetch_array($result))
                {
                print ("b>You are not authorized!</b>");
                <...>
                exit;
```

```
                }
          else
                    {$datetime = date("Y-m-d H:i:s");
                     $username = $row[username];
                     $username = strtolower($username);
                     $result = mysql_query("UPDATE
adminlog SET time = '$datetime' WHERE userid =
'$sessionid'");}
```

Mit den Anweisungen im else-Block wird die Zeit des letzten Zugriffes aktualisiert. So kann das System (unabhängig vom Verfallsdatum des Cookies) feststellen, wie lange ein Benutzer auf einer Seite verblieben ist und eventuell bei zu langer Inaktivität (d.h. Benutzer hat die Seite verlassen, ohne sich abzumelden) die Session beenden.

Da diese Operationen vor jedem Seitenaufruf ausgeführt werden müssen, wurden sie in der Datei check.php gespeichert. Jede Seite, für die eine Überprüfung des Benutzer stattfinden soll, braucht diese nur noch mittels

```
<?php require_once('./check.php'); ?>
```

einzubinden.

4.8.3. Authorisation

Damit die soeben beschriebene Cookie-Authentifizierung funktioniert, muss natürlich ein solches auch beim Anwender gesetzt werden. Dies geschieht zu Anfang einer Session. Dieses Skript heißt auth.php, Benutzername und Passwort werden vom Login-Formular übergeben. Daraufhin werden folgende Funktionen ausgeführt:

```
if(isset($un)and isset($pw))
{

$mysql_link = mysql_connect("localhost","dira","pw") or
die ("Failed to connect to host!");
        mysql_select_db("dira", $mysql_link) or die
("Failed to connect to database!");
        $result = mysql_query("SELECT * FROM admins
                        WHERE login = '$un'
                        and password = '$pw' ");

…
}
```

Nach der Überprüfung, ob die Variablen $un (für Username) und $pw (für Passwort) vorhanden sind (isset()), verbindet sich das Skript mit der Datenbank und sucht nach Datensätzen in der Tabelle admins, bei welchen der Benutzername und das Passwort übereinstimmen.

An dieser Stelle sei noch auf eine Gefahr dieses Quelltextes hingewiesen: Falls auf dem Server die Option magic_quotes_gpc nicht aktiviert ist, kann ein böswilliger Benutzer die Passwortabfrage leicht umgehen. Falls er etwa als Benutzername admin'# eingibt, so würde die SQL-Anfrage folgendermaßen lauten:

```
SELECT * FROM admins WHERE login = 'admin'#'
                        and password = '$pw'
```

Da alle Zeichen hinter der Raute (#) als Kommentar gelten, würde die Anfrage auch ohne Passwort für den Benutzer admin einen Treffer ausgeben. Der Schalter magic_qotes_gpc (in der Konfigurationsdatei für *PHP*) sorgt dafür, dass in allen GET und POST-Anfragen sowie Cookies Sonderzeichen (wozu auch ' gehört) mittels eines vorgestellten Backslashes (\) „entschärft“ werden[123,124].

Das auth.php Skript überprüft dann im Folgenden, ob ein Treffer vorliegt. Ist dies nicht der Fall, so wird mittels

---

[123] http://de.php.net/features.file-upload

[124] [ANO1999], S. 422 - 430

```
$message="Authorisation fehlgeschlagen: falscher
Username oder falsches Password!";

header("Location:../index.php?message=$message");
```

eine Umleitung auf die Startseite (mit entsprechender Fehlermeldung) durchgeführt. Wird ein Datensatz gefunden, so unterscheidet das Skript zwischen zwei Fällen:

- Der Benutzer ist noch nicht authentifiziert: Das Anfangs erzeugte Cookie wird mittels der Query

```
INSERT INTO adminlog (username, userid,time,hostname)
  VALUES ('$un','$sessionid','$datetime','$hostname')
```

  in die Tabelle adminlog eingetragen.

- Der Benutzer ist bereits authentifiziert: Dieser Fall tritt ein, wenn ein Benutzer die Webseite zuvor verlassen hat, ohne sich ordentlich abzumelden. Die Query lautet dann:

```
UPDATE adminlog SET userid = '$sessionid', time =
'$datetime', hostname = '$hostname' WHERE adminlogid =
'$rows[adminlogid]'
```

  Hiermit wird der bestehende Eintrag für den Benutzer aktualisiert.

Diese Unterscheidung hat zur Folge, dass nicht mehrere Benutzer mit derselben Kennung gleichzeitig auf dem System arbeiten können. Das ist auch gewollt, da es keinen Sinn macht, wenn zwei unterschiedliche Anwender zur gleichen Zeit das System umkonfigurieren. Sind diese Abfragen durchgeführt, wird die Startseite des geschützten Bereiches (adminindex.php) geladen.

4.8.4. Accounting

Dieser Teil von AAA ist nur ansatzweise implementiert. Es werden alle vom Benutzer ausgeführten Aktionen in ein Protokoll gespeichert, so dass daraus bei Bedarf leicht eine Abrechnung erfolgen kann. Für die derzeitige Nutzung der Software ist dies aber noch nicht erforderlich.

4.8.5. Die Benutzeroberfläche

Die Benutzeroberfläche stellt wie bereits erwähnt die einzige Möglichkeit der Interaktion des Benutzers mit dem System dar. Sobald sich ein Benutzer mit dem System verbunden hat, präsentiert sich ihm folgende Menüstruktur

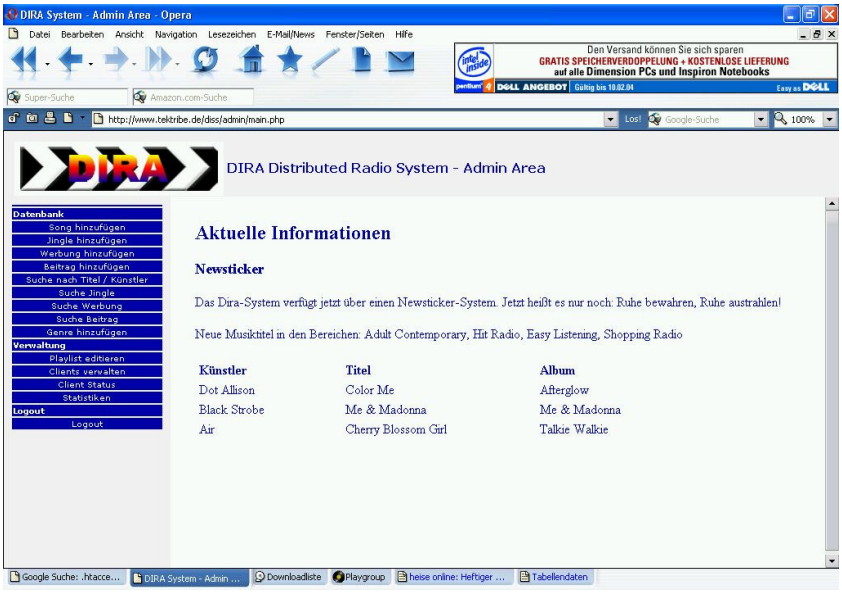

Abbildung 4.8.5a – Hauptmenü des DIRA-Systems

Die Gestaltung entspricht den gängigen Standards bei Webseiten. Durch Klicken auf das DIRA-Logo links oben in der Ecke gelangt man immer zur Startseite zurück. Auf dieser werden mit einem simplen Content Management System (CMS) aktuelle Informationen sowie neu hinzugefügt Musikstücke dargestellt. Die Seite lädt sich automatisch jede Minute neu, so dass die dargestellten Informationen immer aktuell

sind. Im linken Bereich befinden sich die verfügbaren Administrationsbereiche.

4.8.6. Datenbank

Im ersten Segment des Administrationsbereiches befinden sich Menüpunkte, die der Verwaltung des Datenbestandes dienen. Dabei hat der Benutzer die Möglichkeit, Musikstücke, Jingles, Beiträge und Werbung zu übertragen, danach zu suchen sowie die vorhandenen Genres zu editieren.

Im Einzelnen handelt es sich dabei um:

- Song hinzufügen: Hier können neue Musikstücke in das System eingespielt werden. Dabei sind neben den offensichtlichen Angaben wie Künstler oder Titel folgende Punkte auszufüllen
    - GEMA: Falls eine Abrechnung mit der Gesellschaft für musikalische Aufführungs- und mechanische Vervielfältigungsrechte[125] gewünscht wird, so ist der einmalige GEMA-Schlüssel, der an jeder verkauften CD zu finden ist, hier einzugeben
    - Geschlecht: Hier sollte das Geschlecht der (vermuteten) Zielgruppe eingetragen werden. Im professionellen Radiobetrieb wird darauf geachtet, eine Mischung aus Titeln, die eher Männern und solchen, die eher Frauen gefallen, zu finden. Unter welche Rubrik ein Lied fällt, ist allerdings Geschmacks- und Erfahrungssache des zuständigen Musikredakteurs.
    - Genre: Um eine noch feinere Unterscheidung zwischen den Titeln als nur durch die Formate (siehe nächster Punkt) zu ermöglichen, ist es möglich, zwei verschiedene Genres für ein Lied festzulegen.
    - Format: Das DIRA-System verfügt standardmäßig über vier verschiedene Formate, welche (abgesehen vom Shopping Radio) den üblichen Radioformaten entsprechen. Ein Titel kann in bis zu vier

[125] http://www.gema.de/

dieser Rubriken aufgenommen werden. Die Formate können folgendermaßen charakterisiert werden:

- Easy Listening: Hier sollen sanfte Instrumentaltitel sowie ruhige Evergreens einen beruhigenden Klangteppich erzeugen[126].
- Adult Contemporary: Aktuelle Hits vermischt mit bekannten Songs der letzten Jahrzehnte. Melodische Titel mit starker Orientierung am Massenmarkt[127].
- Shopping: Hier handelt es sich nicht um ein bestehendes Format im herkömmlichen Sinne. Da der Einsatzbereich des DIRA-Systems auch im Bereich der Beschallung von Verkaufsflächen liegen könnte, bezeichnet dieser Begriff ein Format, welches zwischen Adult Contemporary und Easy Listening liegt.
- Hit Radio: Dieses Format wird manchmal auch Contemporary-Hit-Radio bezeichnet. Es werden hauptsächlich aktuelle Hits gespielt, diese teilweise in sehr kurzen Wiederholungsintervallen. Man findet kaum Titel, die älter als 6 Monate sind[128].

- Datei: Die zu übertragende Datei.

Sind diese Felder ausgefüllt, so überträgt der Webbrowser des Benutzers die Daten der ausgewählten Datei mittels der POST-Methode an den Server. Es sei noch erwähnt, dass die Punkte Genre, Geschlecht und Geschwindigkeit noch nicht zur individuellen Gestaltung der Playlist genutzt werden können. Dies ist erst in einer späteren Version des Systems vorgesehen. Um sicherzustellen, dass die transferierten Stücke fehlerfrei und im richtigen Format sind, wird nun eine Reihe von Hilfsprogrammen durch den Webserver aufgerufen. Dabei handelt es sich um:

---

[126] http://www.radiomanager.de/Formate.htm

[127] http://www.rms.de/order_check/download/angebote/radioformate.pdf

[128] http://www.radioszene.de/chr.htm

- *ismp3*: Dieses Skript überprüft die Datei mit Hilfe des *mp3_check*-Programms und ermittelt, wie viele fehlerfreie Frames (Good Frames) vorhanden sind. Ist dieser Wert Null, so wird davon ausgegangen, dass es sich um keine MP3-Datei handelt und ein Errorlevel von 0 (entsprechend false) zurückgegeben.
- *is128*: Hiermit wird die Ausgabe von *mp3_check* im Bezug auf die Bitrate untersucht. Beispielhaft für die anderen Hilfsprogramme eine kurze Beschreibung des Shell-Skriptes:

```
$string = qx(/usr/local/bin/mp3_check -v
„@ARGV[0]“ 2>&1 | grep „BIT_RATE“)
```

Die Ausgabe von *mp3_check* (ausgeführt durch den qx() Befehl) wird mittels einer Pipe an *grep* geleitet, welches alle Zeilen, die nicht den String BIT_RATE enthalten ausfiltert. In der verbleibenden Zeile werden daraufhin mittels eines regulären Ausdruckes alle alphanumerischen Zeichen sowie Sonderzeichen entfernt:

```
$string =~ s/[A-Z,_, ,.,:]//g;
      if ($string < 128)
  {exit(0);} else {exit(1);}
```

Ist der verbliebene String kleiner als 128, so beendet das Skript mit einem Errorlevel von Null. Das Ziel dieser Überprüfung ist es, keine MP3-Dateien mit einer kleineren Bitrate als 128 zuzulassen. Zwar könnte der später folgende Encoder die Bitrate etwa einer 64 KBit/s Datei auf 128 KBit/s hochrechnen, die Klangqualität würde aber auf Grund der verlustbehafteten Funktion des MP3-Verfahrens leiden.

- *is44*: Der folgende Encoder kann kein Downsampling von 48000 kHz auf 44100 kHz betreiben. Daher werden durch dieses Skript

Dateien mit einer zu hohen Sampleauflösung ausgefiltert. Die Funktionsweise ist analog zum is128 Skript.

- *isstereo*: Eine weitere Einschränkung des Encoders ist, dass er Mono-Dateien nicht in das Stereo-Format überführen kann. In diesem Abschnitt findet daher eine Überprüfung der Datei auf Stereo-Eigenschaften statt.
- *mp3check*: Hier erfolgt das Neukodieren der Datei. Mittels des *Lame*-Encoders konvertiert das *mp3check.pl*-Skript die Daten in eine 128 KBit/s MP3-Datei im Joint-Stereo Format.
- *mp3gain*: Abschließend wird dieses Programm (es ist als einziges kein Skript) aufgerufen um, wie bereits beschrieben, die Lautstärke der Titel anzupassen.

Liefert eines der eben genannten Skripte einen Errorlevel von 0 zurück, so wird die Bearbeitung abgebrochen und eine entsprechende Fehlermeldung ausgegeben. Abschließend wird nun der Dateiname, unter welchem die Daten auf dem Server gespeichert werden, angepasst. Das ist notwendig, da Windows andere Zeichen in Dateinamen erlaubt als Linux. Ein Beispiel hierfür wäre der Schrägstrich (/), der unter Linux ein Verzeichnis beschreibt, unter Windows aber im Dateinamen vorkommen darf. Folgende Zeichen werden gefiltert:

/ \ ` ´ , # & ;

Es wäre vermutlich einfacher, auf die Verwendung des Originaldateinamens zu verzichten und stattdessen einen numerischen Wert zu benutzen. Um aber die Funktion des Systems besser überprüfen zu können (ein Dateiname wie jingle1-test.mp3 ist aussagekräftiger als 000023.mp3), habe ich mich für diese Lösung entschieden.

Nach diesen teilweise recht umfangreichen Operationen (sie können je nach Dateigröße bis zu etwa 1 Minute dauern) kopiert das PHP-Skript die Datei in

den vorgesehenen Ordner /home/dira und trägt die vom Benutzer angegebenen Meta-Informationen in die *MySQL*-Datenbank ein. Ebenfalls wird dort der neue Dateiname inklusive Verzeichnis gespeichert.

- Jingles, Werbung bzw. Beitrag hinzufügen: In diesen Bereichen kann der Anwender die entsprechenden Programmelemente hinzufügen. Die Upload-Routinen sind mit denen des Musik-Uploads identisch. Folgende Felder sind auszufüllen:
    - Sendung (bzw. Werbekunde oder Beitrag): Der Titel des Elementes, wie er in der Archivdarstellung angezeigt wird.
    - Jingle (bzw. Spot oder Thema): Hier ist eine genauere Beschreibung möglich.
    - Dateiname: Wie beim Musik-Upload muss der Anwender eine Datei auf seiner lokalen Festplatte auswählen.
    - Wiederholungen: Hier kann festgelegt werden, wie häufig ein Element gespielt wird. Die hier angegebene Zahl bezieht sich auf die Wiederholungen pro Client – da keine Kommunikation zwischen den Clients vorgesehen ist, wissen diese nicht, wie oft ein Titel global (also auf allen Rechnern) wiederholt wurde.

  Nach dem Upload werden die Dateien in die Verzeichnisse /home/dira/ads, /home/dira/features bzw. /home/dira/jingles kopiert und die angegebenen Metadaten in die Datenbank eingetragen.

- Suche nach Künstler/Titel: Hiermit ist die Verwaltung der vorhandenen Musiktitel möglich. Mit der Eingabemaske kann nach dem Künstlernamen oder dem Titel eines Liedes gesucht werden. Weiterhin ist eine Einschränkung nach dem Format möglich. Will man sich alle Titel anzeigen lassen, so genügt das Drücken der Eingabe-Taste. In der daraufhin angezeigten Liste sind zwei Funktionen pro Titel möglich:

- Ändern: Hiermit können die Meta-Informationen (etwa Genre, Format, Künstlername etc.) geändert werden. Wie diese Information an die Clients übertragen wird, ist in Abs. 4.12 genauer beschrieben.
- Löschen: Hiermit wird der entsprechende Titel aus der Datenbank entfernt, die korrespondierende MP3-Datei wird gelöscht. Vorher wird eine Sicherheitsabfrage durchgeführt.

- Suche Jingle, Werbung bzw. Beitrag: Analog zur Musiktitelverwaltung kann hier der Programmelementbestand verwaltet werden. Die Bedienung ist identisch zum Menüpunkt „Suche nach Künster/Titel".
- Genre hinzufügen: Hier kann der Anwender beliebig viele Genres definieren, nach denen in zukünftigen Versionen eine differenzierte Programmgestaltung möglich sein wird.

4.8.7. Verwaltung

In diesem Abschnitt des Menüs befinden sich verschiedene Punkte zur Administration der Clients sowie zur Gestaltung der Playlist:

- Playlist editieren: Hiermit ist eine Verwaltung der für alle Clients gültigen Playlist möglich. Da ebenso wie beim TOX-System eine minuten- oder stundengenaue Programmgestaltung nicht vorgesehen ist, kann unter dem Feld „Segmente in der Programmuhr" die Anzahl der Elemente pro Wiederholung festgelegt werden. Dadurch ist es dem Anwender möglich, den Ablaufplan für etwa eine Stunde zu erstellen. Die Zahl der benötigten Elemente hängt von der Dauer der Musikstücke sowie der Jingle-, Werbe- Beitragselemente ab. In der Regel sollte ein Wert zwischen 14 und 18 in etwa eine Stunde abbilden.

  Im Weiteren kann für jeden Programmplatz ein Element ausgewählt werden. Für den Fall, dass die Spalte Typ den Wert „Musik" enthält, kann unter Format / Titel ein entsprechendes Format ausgewählt werden. Die Planung

eines bestimmten Musiktitels ist nicht vorgesehen – dies würde auf Grund der stündlichen Wiederholung auch keinen Sinn machen.

Entspricht Typ Jingles, Werbung oder Beitrag, so hat der Anwender die Möglichkeit, ein spezielles Element für diesen Programmplatz zu planen. Ist dies nicht gewünscht, so kann er unter Format / Titel den Wert „beliebig“ einstellen, die Clients spielen dann einen zufällig ausgewählten Titel aus dem Archiv.

- Clients verwalten: Dieser Menüpunkt dient der Administration der angeschlossenen Clients. Unter dem Punkt „Neuen Client erstellen“ kann man die Datenbank so erweitern, dass ein neuer Client (der natürlich auch in Form eines Rechners vorhanden sein muss) darauf zugreifen kann. Dabei ist es möglich, für jeden Client unterschiedliche Formate festzulegen. Alternativ gibt es die Möglichkeit, Clients zu löschen, was zur Sicherheit noch einmal abgefragt wird.
- Client Status: Dieser Menüpunkt dient der Überwachung der Client-Aktivitäten. Da am Gerät selbst keine Bedienoberfläche vorgesehen ist, ist es notwendig, Status- und Fehlermeldung auf der Webseite zugänglich zu machen. Im rechten Bildschirmbereich ist eine Liste der angeschlossenen Clients mit dem jeweiligen Zeitpunkt des letzten Updates zu sehen. Clients, welche länger als 24 Stunden kein Update durchgeführt haben, werden dabei rot markiert. Durch Klicken auf den Info-Link werden spezifische Informationen zum entsprechenden Client angezeigt. Diese beinhalten unter anderem die vom Client gesendeten Log-Nachrichten. Dabei werden nur solche Nachrichten angezeigt, deren Urgent Feld 1 beträgt (siehe Abs. 4.9.5). Dadurch soll vermieden werden, dass der Anwender durch zu viele Einträge verwirrt wird.
- Statistiken: Dieser Punkt ist ähnlich wie „Client Status“ aufgebaut. Ruft man ihn auf, so erscheint eine Übersicht über die Anzahl der Wiederholungen der vorhandenen Jingle-, Werbung- und Beitrag-Elemente. Die Zahlen beziehen sich dabei darauf, wie häufig ein Element insgesamt auf allen Clients gelaufen ist. Weiterhin ist es möglich, sich mittels des Info-

Links die Daten eines einzelnen Clients anzeigen zu lassen.

4.8.8. Abmeldung

Hier kann die aktive Sitzung eines Benutzers beendet werden. Dabei wird das Cookie aus der Datenbank gelöscht. Dies bewirkt, dass es nicht mehr möglich ist, mittels der „Zurück“ Funktion des Webbrowsers auf die Seite zurückzukehren. Vor allen an Rechnern mit mehreren verschiedenen Benutzern (z.B. Internetcafé) ist es daher sinnvoll, sich immer mit Logout abzumelden.

4.9. Die MySQL-Datenbank

In der *MySQL*-Datenbank werden die Meta-Informationen sowie die Dateinamen der vorhandenen Titel gespeichert. Zusätzlich werden dort eine Reihe von Konfigurations-, Status- und Nachrichteninformationen abgelegt.

Damit die Client-Software auf die Datenbank des Servers zugreifen kann, ist es notwendig, die auf dem Server laufende Software entsprechend zu konfigurieren. Standardmäßig wird *MySQL* über das Skript mysql im Verzeichnis /etc/init.d beim Hochfahren gestartet. Um den Server dahingehend zu konfigurieren, dass er auch auf einer externen Netzwerkschnittstelle auf Verbindungen wartet, muss man den Schalter –bind-address=127.0.0.1 löschen. Dieser bewirkt, dass die Datenbank nur über das Loopback-Device (eine Art interne Netzwerkkarte) erreichbar ist.

Im Weiteren sollen nun die wichtigsten Tabellen innerhalb der Datenbank beschrieben werden.

4.9.1. Das Archiv

Für die Archivierung sowohl der Musik- als auch der Programmelementdaten kommen vier verschiedene Tabellen zum Einsatz, welche sich recht ähnlich sind und daher in einem Kapitel zusammen behandelt werden. Die Namen der Tabellen lauten:

- archive
- ad_archive
- jingles_archive
- feature_archive

Die Tabelle archive setzt sich zum Beispiel aus folgenden Spalten zusammen:

- counter: Dieser Zähler wird für jeden neuen Eintrag automatisch erhöht (auto_increment) und ist der Primärschlüssel, über welchen *MySQL* einen Datensatz schnell ansprechen kann.
- artist, songtitle, album, release, gender, speed, gema, style, style1: In diese Felder werden die Daten aus dem Formular zum Hinzufügen von Titeln gespeichert. Die Namen der Spalten sind auf Englisch gehalten, release bezeichnet das Veröffentlichungsjahr, style bzw. style1 stehen für das Genre. Die weiteren Felder dürften selbsterklärend sein.
- ac, hit, easy, shop: Diese Felder können nur die Werte 0 oder 1 annehmen (sie sind als tinyint(1) definiert) und bezeichnen das Format eines Titels. Somit kann ein Titel zugleich in bis zu vier verschiedene Formate eingeordnet werden.
- filename: Dieser Wert bezeichnet den Dateinamen auf dem Server. Auf Grund der in Abs. 4.8.6 beschriebenen Bearbeitung entspricht er nicht mehr dem vom Benutzer angegeben Namen. Dieser Schlüssel ist in der Datenbank als „unique“ gekennzeichnet. Das bedeutet, dass er nur einmal vorkommen

darf – das erhöht zum einen die Geschwindigkeit der *MySQL*-Abfragen[129], zum anderen bildet dies das Verhalten des Dateisystems (es können keine zwei Dateien denselben Namen haben) ab.

- play, bitrate, samplerate, playtime, uploader, upload: Diese Werte werden ebenfalls vom Server automatisch ausgefüllt. Sie bezeichnen die Anzahl der Wiederholungen (play), die Bit- und Samplerate, die Spieldauer (playtime), den Namen des Uploaders sowie den Zeitpunkt des Uploads (upload). Dieser ist für die Synchronisation des Datenbestandes mit den Clients wichtig, wie wir folgenden Abschnitt sehen werden.

In den Tabellen für die weiteren Programmelemente (ad_archive, jingles_archive, feature_archive) wurde auf die Spalten album, release, speed, gender, gema, style1, ac, hit, shop und easy verzichtet, da sie dafür nicht benötigt werden. Stattdessen wurde die Spalte play_limit hinzugefügt – sie bezeichnet die maximale Anzahl der Wiederholungen eines Elements.

4.9.2. Die Synchronisationstabellen

Auf Grund der verteilten Architektur des Systems sind einige Vorsorgungen bezüglich der Synchronität des Datenbestandes von Server und Clients zu treffen. Wird etwa ein Titel auf dem Server gelöscht, so wird auch der entsprechende Eintrag in der Datenbank gelöscht. Um festzustellen, dass ein Titel gelöscht wurde, müsste der Client alle Einträge in seiner lokalen Datenbank mit dem Server abgleichen, was bei einem umfangreichen Archiv relativ lange dauern kann. Intelligenter ist es, in einer separaten Tabelle festzuhalten, welche Titel gelöscht (oder analog dazu: geändert) wurden, die Clients können diese Aktion dann einfach nachvollziehen. Hierfür kommen zwei Tabellen zum Einsatz:

- deleted: Sie besteht aus den Schlüsseln counter, filename und deleted. Letzterer gibt dabei den Timestamp des Löschvorgangs an. Der Client sucht

[129] http://www.mysql.com/doc/de/Indexes.html

in der lokalen Datenbank nach dem entsprechenden Dateinamen und löscht diesen.

- updated: Damit Änderungen im Datenbestand (z.B. Änderung des Formats eines Titels) auch an die Clients weitergegeben werden, überprüfen diese bei einem Update, welche Elemente und Musikstücke verändert wurden. Updated besteht dabei aus den Schlüsseln counter, updateid, updated und type. Updateid bezeichnet den Counter des entsprechenden Elements, updated ist der Timestamp und type gibt an, ob es sich um ein Musikstück oder Programmelement handelt.

4.9.3. Die Konfigurationstabellen

Unter diesem Begriff werden verschiedene Tabellen zur Speicherung von Konfigurationsdaten zusammengefasst.

- admins: In dieser Tabelle (bestehend aus den Spalten login, password und name) werden die erlaubten Benutzer des Systems gespeichert. Sie wird zur Autorisierung im auth.php Skript genutzt. Das Feld login ist der Primärschlüssel, daher kann derselbe Name nur einmal gewählt werden.
- adminlog: Hier verwaltet das System die momentan angemeldeten Benutzer. Dazu speichert es das 32-stellige Cookie neben dem Benutzernamen und der IP-Adresse. Auf diese Tabelle wird vor jedem Seitenaufruf zugegriffen, um festzustellen, ob der Benutzer berechtigt ist, die Seite zu sehen.
- config: In dieser Tabelle werden die angeschlossenen Clients eingetragen. Dabei werden folgende Schlüssel benutzt:
    - client: Die Nummer des entsprechenden Clients. Dies ist zugleich der Primärschlüssel der Tabelle
    - last_update: Dieser Timestamp gibt an, wann der entsprechende Client das letzte Mal ein Update durchgeführt hat.
    - ac, hit, easy, shop: Für jeden Client kann hiermit festgelegt werden,

aus welchen Formaten er Titel spielen soll.

- city, street: Zur einfachen Identifikation durch den Benutzer kann hier der Standort des Clients eingetragen werden.

Anzeige und Änderung dieser Informationen erfolgt ausschließlich über die in Abb. 4.8.3 beschriebene Clientverwaltung.

4.9.4. Die Playlisttabellen

Das DIRA-System beherrscht den Umgang mit individuellen Vorgaben an die zu spielenden Titel. Dazu werden zwei verschiedene Tabellen benutzt:

- planned_songs: Hier kann für jeden Client die Anzahl der Titel pro Durchlauf festgelegt werden. Die Tabelle besteht aus den zwei Schlüsseln client und clientsongs.
- planned_songs_list: Hier werden die Informationen bezüglich der einzelnen Positionen der Playlist gespeichert. Die Tabelle ist standardmäßig mit 20 Einträgen gefüllt. Wie viele Elemente letztendlich gespielt werden, wird durch den Wert in planned_songs festgelegt. Im Einzelnen besteht die Tabelle aus:
  - counter: Bestimmt die Reihenfolge der Einträge und ist zugleich Primärschlüssel
  - type: Enthält entweder Music, Jingles, Ad oder Feature und bestimmt so, ob der Eintrag ein Musikstück oder ein Programmelement bezeichnet.
  - specific: Bei Programmelementen kann hier der Counter eines speziellen Elements (z.B. ein ausgewählter Werbespot) gesetzt werden. Ist type Music, so ist dieser Schlüssel immer NULL.
  - style, speed, gender: Hiermit kann bei Musikstücken das Genre, die Geschwindigkeit oder die Zielgruppe eingeschränkt werden.

- Format: Dieser Schlüssel gilt nur für Musiktitel. Die möglichen Einträge lauten hier ac, hit, easy und shop und legen das gewünschte Musikformat des gewählten Eintrages fest. Handelt es sich dabei um ein Programmelement, so ist der Wert immer NULL.

4.9.5. Weitere Tabellen

Zusätzlich zu den beschriebenen Tabellen benutzt das DIRA-System eine Reihe weiterer Strukturen für verschiedene Aufgaben. Dabei handelt es sich um:

- Serverprotokolle: In der Tabelle log werden die auf dem Server durchgeführten Aktionen protokolliert. Ändert ein Benutzer etwa die Playlist oder fügt er neue Titel hinzu, so wird dies im Serverprotokoll gespeichert. Somit lassen sich Änderungen oder Fehlbedienungen nachvollziehen. Folgende Werte werden gesichert:
  - logtime: Der Zeitpunkt der Aktion
  - action, hostname: Beschreibt die durchgeführte Aktion und von wo sie gestartet wurde.
  - origin: Unter welchem Benutzername wurde die Aktion gestartet

  Ein Beispiel für einen solchen Eintrag wäre folgende Zeile:

Abbildung 4.9.5a – Auszug aus dem Serverprotokoll

- Clientprotokolle: Da die Clients über keine Benutzerschnittstelle verfügen, ist es nötig, eventuelle Fehlermeldung auf den Webseiten darzustellen. Gibt es bei der Abarbeitung auf den Clients ein Problem, so wird dieses zuerst lokal gespeichert und beim nächsten Update an den Server übertragen. Die hierfür zuständige Tabelle nennt sich clientlog und besteht aus folgenden Feldern:

- counter: Der Primärschlüssel zum Indizieren der Tabelle
- clientid: Die ID-Nummer des protokollierenden Clients
- time: Der Zeitpunkt der Fehlermeldung
- message: Die aufgetretene Fehlermeldung
- urgent: Dieser Schlüssel kann nur 0 oder 1 annehmen. 1 bezeichnet eine schwerwiegendes Problem, 0 ein geringfügiges

Die Meldung können unter dem Menüpunkt „Client Status“ angezeigt werden.

- Nachrichtenticker: Um auf der Startseite des Benutzerbereiches aktuelle Informationen anzeigen zu können, müssen diese in der Tabelle news gespeichert sein. Sie besteht aus den Einträgen datetime, text und who, welche das Datum, die Nachricht und den Verfasser bezeichnen
- Statistiken: Für die Erzeugung der statistischen Auswertung wird die Tabelle statistics benutzt. Sie besteht aus den bekannten Schlüsseln counter, clientid, element, artist, songtitle, filename und play. Diese Tabelle wird beim Update der Clients beschrieben, für die Webseite kommen zwei verschiedene Queries zum Einsatz:
  - Um die statistischen Daten für alle Clients anzuzeigen, benutzt der entsprechende Menüpunkt folgende Query:

```
SELECT artist,songtitle,SUM(play),filename FROM
        statistics GROUP BY filename
```

    Dadurch wird die Anzahl der Wiederholungen aller Clients mittels SUM(play) aufaddiert, zusätzlich werden die Einträge nach Dateinamen gruppiert (GROUP BY filename).
  - Für die Darstellung der Daten eines einzelnen Clients lautet die Query:

```
SELECT artist,songtitle,play FROM statistics WHERE
        client=$client
```

Sie bewirkt, dass nur Einträge des gewählten Clients benutzt werden.

4.10. Die System-Schnittstellen

Eingebettet zwischen die Beschreibung der Server- und den folgenden Clientkomponenten soll hier noch genauer auf die weiteren Schnittstellen eingegangen werden. Die in Abs. 4.5.3 erklärten Webseiten stellen dabei das Benutzer-Interface dar, zusätzlich dazu gibt es zwei interne Systemschnittstellen. Sie dienen ausschließlich der Kommunikation zwischen Client und Server, ihre Funktionsweise ist für den Benutzer transparent. Im Einzelnen handelt es sich dabei um:

4.10.1. MySQL-Schnittstelle

Damit die Clients auf die serverseitig gespeicherten Informationen zugreifen können ist es notwendig, den Datenbankzugriff über das Netzwerk zu erlauben. Um zu verhindern, dass Unbefugte dadurch Zugriff auf das System erhalten, gilt es, das „Access Control List“ (kurz ACL) Modul von *MySQL* zu nutzen. Dabei handelt es sich um Tabellen, welche die Zugriffsrechte auf die Datenbank verwalten.

- User-Rechte: Es ist sinnvoll, für den Zugriff durch die Clients einen neuen Benutzer mit stark eingeschränkten Rechten anzulegen. Für das DIRA System wurde dabei eine Deny-All Strategie benutzt. Das bedeutet, dass der *MySQL*-Benutzer (welcher komplett unabhängig von einem System-Benutzer ist) mit dem Namen „Client“ grundlegend keinerlei Rechte hat, irgendwelche Daten einzusehen oder zu manipulieren.
- Datenbank- bzw. Tabellen-Rechte: Da so natürlich kein Zugriff möglich wäre, gilt es „Löcher“ in diese sehr strikte Regel zu schneiden. Dabei ist darauf zu achten, dass die Zugriffsrechte möglichst straff an den

Anforderungen der Clients organisiert sind. Folgende Einstellungen kommen beim DIRA-System zum Einsatz:

- Datenbank dira: Der Benutzer client darf nur Daten aus dieser Datenbank lesen (SELECT).
- Tabelle statistics: Bei dieser Tabelle ist zusätzlich das Einfügen sowie das Aktualisieren von Daten erlaubt (INSERT bzw. UPDATE). Dies wird benötigt, damit die Clients die statistischen Auswertungen auf den Server übertragen können.
- Tabellen config, ad_archive, jingles_archive, feature_archive: Für diese Tabellen ist nur das UPDATE-Recht erforderlich, damit bei den Programmelementen die Anzahl der Wiederholungen bzw. der Zeitpunkt des letzten Updates aktualisiert werden können.

Die Clients erhalten so keine Möglichkeit, Datensätze zu löschen oder das Archiv zu verändern. Dies dient zur zusätzlichen Absicherung, sollte das *MySQL*-Passwort für einen Benutzer bekannt werden.

Leider findet sämtliche Kommunikation unverschlüsselt statt, so dass es mit Hilfe eines Netzwerk-Sniffers möglich wäre, das Passwort auszulesen. In Abs. 5.2 wird eine Gegenmaßnahme gegen diese Schwachstelle diskutiert. Ein solcher Angriff setzt aber Zugriff auf die Netzwerk-Infrastruktur voraus und ist somit in der Regel nur durch den Betreiber des Clients durchführbar.

4.10.2. Die FTP-Schnittstelle

Um einen unkomplizierten Zugriff auf die auf dem Server gespeicherten Musik-Dateien zu ermöglichen, habe ich mich für die Benutzung der FTP-Schnittstelle entschieden. Da das File Transfer Protocol eines der ersten entwickelten Transferprotokolle des Internets ist, finden sich entsprechend viele Programme und Zusatzmodule für alle nur erdenklichen Sprachen.

Damit die Clients sich Titel von Server übertragen können, muss zuerst ein entsprechender Systembenutzer eingerichtet werden. Der entsprechende Eintrag in der Datei /etc/passwd sieht dabei so aus:

```
client:x:508:49::/home/dira:ftp
```

Durch die Angabe von ftp als Shell (letzter Wert) kann er keine Terminal-Sessions starten, sondern erhält nur Zugriff per FTP. Dabei wird auch von der Chroot-Konfiguration gebraucht gemacht: Da als Home-Verzeichnis /home/dira angegeben ist, kann der Client nur auf dieses Verzeichnis sowie vorhandene Unterverzeichnisse zugreifen. Man spricht dabei von einem geändertem Wurzelverzeichnis (Changed Root).

Die eigentliche Steuerung der Transfers geschieht dabei durch Textbefehle. Wird vom Client etwa ein `GET /home/dira/jingles/jingle01.mp3` gesendet, so überträgt der Server die entsprechende Datei an diesen Client. Dabei ist zu beachten, dass vor der Übertragung der richtige Modus gewählt wird: Überträgt man nämlich (binäre) MP3-Dateien im Text Modus, so erhält man auf dem Client stark gestörte oder nicht abspielbare Kopien der Datei.

Zusätzlich zum Transfer der Daten wird noch auf die Size-Funktion zurückgegriffen. Sie gibt die Dateigröße auf dem Server an, dadurch können die Clients bestimmen, ob die Übertragung vollständig abgeschlossen ist.

4.11. Komponenten des Clients

Die Komponenten der Clients greifen wie in Abb. 4.11a dargestellt ineinander:

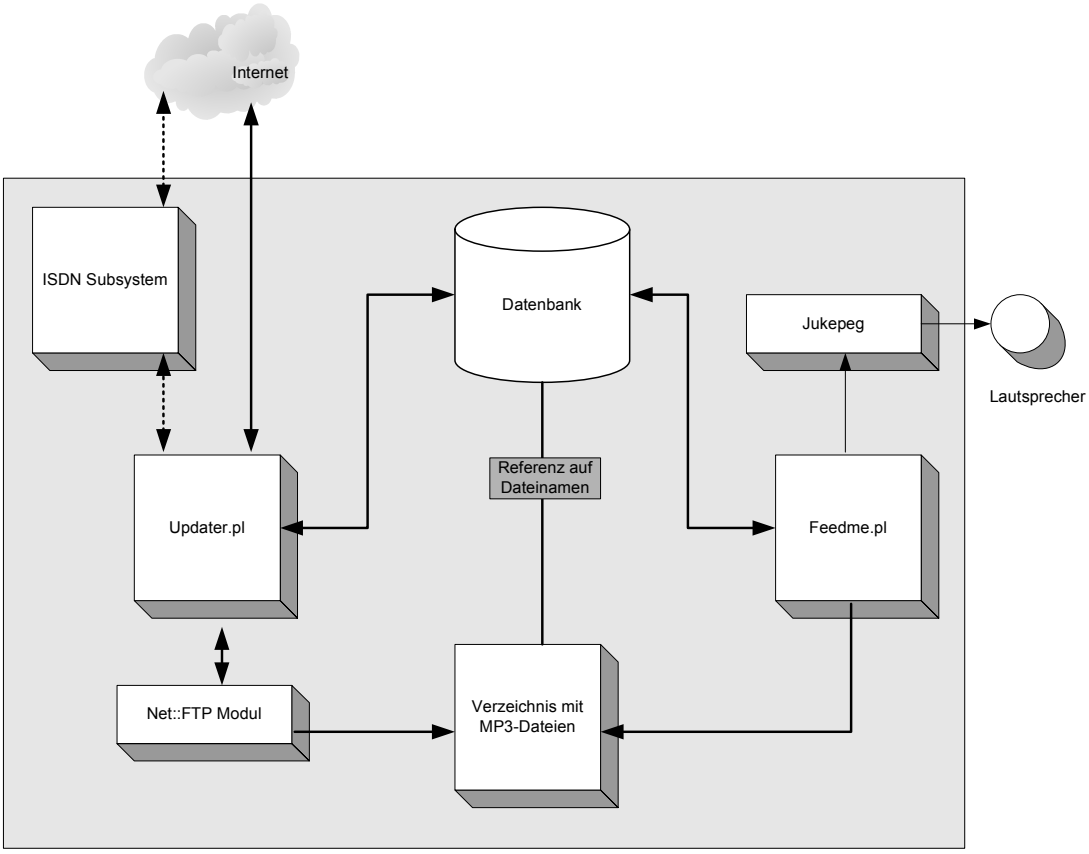

Abbildung 4.11a – Blockschaltbild des Clients

Dabei wird deutlich, dass es neben der lokalen Datenbank noch zwei weitere Komponenten gibt. Dabei handelt es sich um das Update- und das Ausspiel-Modul (*Feedme.pl*), welche beide die Datenbank als Container benutzen. Die Module selber benutzen noch eine Reihe von weiteren Hilfsmodulen, auf die in den entsprechenden Abschnitten genauer eingegangen wird.

4.11.1. Das Update-Modul

Das Update-Modul hat die Aufgabe, den Datenbestand des Clients mit dem Server synchron zu halten. Es wird auf dem Client periodisch durch den *Cron*-Dämon aufgerufen, verbindet sich mit dem Server und gleicht die geänderten Daten ab. Es ist sozusagen das Herzstück des verteilten Systemdesigns und soll daher ausführlich

behandelt werden. In den ersten Versionen der DIRA-Software wurde für diese Replikation der Daten die integrierten Routinen von *MySQL* genutzt[130]. Leider stellte sich heraus, dass diese sich nur eingeschränkt für den Betrieb über eine nicht permanente Internetverbindung eignen, da die Datenbanken mehrmals ohne ersichtlichen Grund die Synchronisation verloren.

4.11.2. Besonderheiten der Synchronisation

Um festzustellen, welche Datensätze neu sind, kommt das Timestamp-Verfahren zum Einsatz. Dadurch ergeben sich allerdings einige Fallen beim Ablauf der Synchronisation, wie im Folgenden deutlich wird:

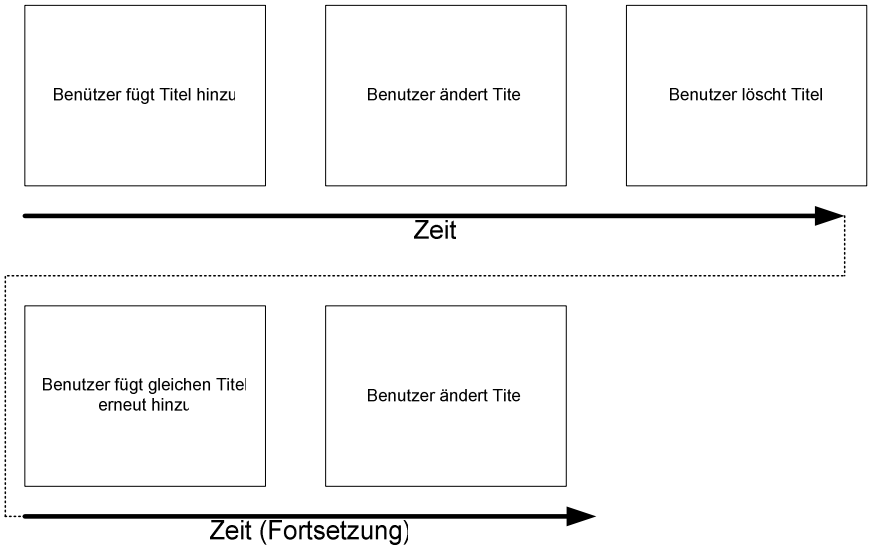

Abbildung 4.11.2a – Beispiel des Zeitablaufes bei Benutzeraktionen

In diesem Beispiel fügt der Anwender einen neuen Titel hinzu, ändert diesen und löscht ihn daraufhin. Anschließend fügt er dieselbe Datei erneut hinzu und ändert sie wieder. In den beteiligten Tabellen archive, updated und deleted finden sich daraufhin folgende Einträge:

[130] http://dev.mysql.com/doc/mysql/en/Replication.html

- Archive: Neuer Titel, Id: y, Zeitpunkt des Uploads x+0
- Updated: Titel geändert, Zeitpunkt x+1, Id des Titels: y
- Deleted: Titel gelöscht, Zeitpunkt x+2, Id: y
- Archive: Neuer Titel, Id: w, Zeitpunkt des Uploads x+3
- Updated: Titel geändert, Zeitpunkt x+4, Id des Titels: w

Da das Update-Modul diese Aktionen aber nicht nach dem Timestamp sortiert ausführt, sondern verschiedene Aktivitäten bündelt (z.B. neue Titel downloaden, gelöschte Titel entfernen), ist folgender Ablauf notwendig, um eine konsistente Datenbank zu erhalten. Dabei gehen wir davon aus, dass der Client das Update erst nach den eben erwähnten Veränderungen durchführt.

1. Gelöschte Titel: Zuerst werden die gelöschten Titel aus der lokalen Datenbank entfernt. Da das Stück mit Id y noch nicht auf dem Client vorhanden ist (der Transfer erfolgt erst später), kann es auch nicht vom Client entfernt werden. Dies ist gewünscht und wird vom Programm toleriert.
2. Neue Titel übertragen: Der zum Zeitpunkt x+3 hinzugefügte Titel wird nun auf den Client übertragen. Der vorherige Titel befindet sich nicht mehr in archive (er wurde ja gelöscht) und wird daher auch nicht transferiert.
3. Updates: Abschließend werden die Änderungen der Titel bearbeitet. Der Update-Eintrag von x+1 wird ignoriert, da die entsprechende Id y nicht mehr im Archiv vorhanden ist.

Durch diesen Ablauf wird sichergestellt, dass zum einen keine Titel unnötig übertragen werden, zum anderen, dass die gespeicherten Informationen auf Server und Client konsistent sind.

Folgender Punkt ist ebenfalls zu beachten: Fügt der Anwender etwa 10 neue Titel

hinzu, so dauert die Übertragung auf den Client natürlich einige Zeit. Wird ein Titel nun während dieser Zeit gelöscht, so entsteht dadurch auf der Seite des Clients eine Fehlermeldung. Dies liegt an der Struktur der Datenbankanfragen:

Durch den $sth->execute Befehl wird ein so genanntes Handle auf die Ergebnisse zurückgeliefert. Der Client liest nun mittels des fetchrow_array-Befehls (er extrahiert aus dem Handle das Ergebnis zeilenweise in ein Array) jeden Eintrag Zeile für Zeile aus und überträgt die entsprechende Datei mittels FTP. Durch das Löschen eines Titels kann es nun vorkommen, dass der vom Client erwartete Datensatz nicht mehr existiert, was zu einer Fehlermeldung führt. Lösen lässt sich dieses Problem mit dem fetchall_arrayref Befehl, welcher alle Ergebnisse in einem 2-dimensionalen Array zurückliefert. Im Unterschied zur ersten Lösung befinden sich alle Ergebnisse bereits auf dem Client und können ohne weiteren Datenbankzugriff abgearbeitet werden. Ein anderer Lösungsansatz wäre der Einsatz einer transaktionalen Datenbank, die solche Probleme durch vorgefertigte Prozeduren umgeht.

4.11.3. Benutzte Subroutinen

Bevor wir den Hauptteil des Programms analysieren, soll noch kurz auf einige häufig genutzte Subroutinen eingegangen werden. Sie behandeln oft verwendete Abläufe wie Fehlerüberprüfung, ISDN-Einwahl usw.

Als erste Funktion bei der Abarbeitung des Skriptes gilt es zu überprüfen, ob es bereits ausgeführt wird. Da es periodisch aufgerufen wird und der Transfer neuer Titel einige Zeit in Anspruch nehmen kann, darf keine zweite Instanz des Programms gestartet werden, welche sämtliche Aktionen dann doppelt ausführen würde. Daher findet man zu Beginn der *Updater.pl* Datei folgenden Code:

```
$lockfile = "/tmp/updater.lock";
if (-e $lockfile) {die "Another Skript is still running
… $!\n";}
```

```
open (“locking”, “>$lockfile”) || die “Cannot open
                    lockfile\n”;
                flock („locking“,2);
```

Damit wird (wie unter Linux üblich) eine Lock-Datei im /tmp Verzeichnis angelegt. Sollte diese bereits vorhanden sein (if (-e $lockfile)), wird die Ausführung abgebrochen.

Da der Autor erst während der Fertigstellung dieser Arbeit einen DSL-Anschluss erhalten konnte, sind alle Routinen des Skriptes auch auf den Betrieb mit einer ISDN-Karte ausgelegt. Die Kontrolle über den ISDN-Kanal erfolgt durch den isdnctrl Befehl. Die Konfiguration, ob ISDN oder eine bestehende Netzwerkverbindung genutzt werden soll, ist in der Variable $isdn gespeichert. Das wird in folgendem Beispiel deutlich:

```
if ($isdn)
{ qx(/usr/sbin/isdnctrl dial ippp0);
sleep 10;
$status = qx(/usr/sbin/isdnctrl status ippp0);
if ($status eq “ippp0 is not connected\n”)
{print „ISDN Einwahl nicht möglich\n“ if $DEBUG;
&bye („ISDN Verbindung konnte nicht hergestellt
werden“,1 }}
```

Ist $isdn definiert (also ungleich 0, eine Besonderheit von *Perl*) wird mittels des qx() Befehls (Quote Executable) isdnctrl mit den Parametern dial ippp0 aufgerufen, was die ISDN-Karte zur Einwahl in das Internet veranlasst. Nach einer 10-sekündigen Wartezeit (das Verbinden und Aushandeln der Verbindungsparameter kann einige Zeit dauern) wird überprüft, ob die Operation erfolgreich verlaufen ist. Dazu wird die Rückgabe von ‚isdnctrl status’ analysiert – entspricht diese „ippp0 is not connected“ so konnte keine Einwahl erfolgen.

Im weiteren Verlauf wird häufig die &check_connection Subroutine aufgerufen. Sie

überprüft, ob die ISDN-Verbindung immer noch besteht – die weitere Ausführung des Skriptes wird abgebrochen, falls dies nicht der Fall ist.

Außerdem ist es notwendig, Fehler bei der Abarbeitung weich abzufangen. Das bedeutet, dass das Skript nicht unkontrolliert abbrechen darf, sondern über eine spezielle Routine folgende Aktionen durchführen muss:

- Logging: Da eine auf dem Client erscheinende Fehlermeldung wegen dessen Blackbox-Character nicht sichtbar ist, muss diese in die lokale Datenbank geschrieben werden. Von dort wird die Meldung an den Server übertragen und ist über die Webseiten einsehbar.
- ISDN Anwahl: Bricht das Skript mit einem Fehler ab, muss die ISDN-Verbindung automatisch getrennt werden, da sonst unnötige Kosten anfallen.
- Lockfile: Damit das Programm beim nächsten Durchlauf wieder startet, muss das Lockfile entfernt werden.

Diese Aktionen werden durch die Funktion &bye durchgeführt. Eine Variante dieser Routine ist &byespecial, welche im Falle des Versagens der lokalen Datenbank durchgeführt wird. Dies ist der fatalste Fehler, da dadurch der Client weder Titel spielen noch Updates durchführen kann. Andererseits läuft der *MySQL* Server extrem stabil, so dass dieser Fall eigentlich nur durch einen Hardware-Defekt (z.B. defekte Festplatte) eintreten kann.

Diese zur Verbindungskontrolle durchgeführten Tests mögen ein wenig übertrieben erscheinen, da aber der Testserver über eine nicht sonderlich zuverlässige Internetverbindung verfügt, war diese erweiterte Fehlerbehandlung nötig.

4.11.4. Initialisierung

Nachdem das Updater-Programm die ISDN-Verbindung hergestellt hat, werden einige Variablen initialisiert. Für den Benutzer von Bedeutung sind dabei:

- $benutzer: Diese Variable wird als Benutzername sowohl für die FTP- als auch *MySQL*-Verbindung benutzt
- $client: Legt fest, welche Client Nummer das Programm benutzt
- $isdn: Ist dieser Wert 1, so wird das ISDN-Subsytem verwendet, ansonsten die normale Netzwerkverbindung (Ethernet)
- $ftppw: Das Passwort für den FTP-Zugang
- $mysqlpw: Das Passwort für den *MySQL*-Zugriff

Im Anschluss daran werden die benötigten Datenbanken verbunden. Dabei werden zwei verschiedene Handles definiert, die jeweils für eine unterschiedliche Datenbank stehen. Die Anweisung

```
$dbhlocal = DBI->connect
(“DBI:mysql:local:localhost”,”root”,””) || &byespecial;
```

weist der Variablen $dbhlocal das Handle für die lokale Datenbank zu (dbh steht als Abkürzung für Database Handle), äquivalent dazu

```
$dbhremote = DBI->connect
(„DBI:mysql:config_$kunde:80.190.100.225“,$benutzer,$mys
qlpw) || &bye („Client kann Verbindung zum Server nicht
herstellen“,1);
```

für die Verbindung zum Server. Die Anweisung hinter dem || Zeichen wird im Falle eines Fehlers durchgeführt, die entsprechenden Subroutinen wurden im vorherigen Abschnitt behandelt.

Für die korrekte Synchronisation ist es erforderlich, dass Server und Client dieselbe Uhrzeit benutzen. Auf dem Client kommt dabei allerdings nicht der ntp-Dämon zum Einsatz, da dieser für eine korrekte Funktion eine permanente Internetverbindung

benötigt (er läuft im Hintergrund und überprüft periodisch, ob die Systemzeit noch korrekt ist). Stattdessen wird auf das ntpdate Programm zurückgegriffen.

```
$ntpstatus = qx(/usr/sbin/ntpdate ntp1.ptb.de
ntp2.ptb.de  tick.fh-augsburg.de tack.fh-augsburg.de);

if ($ntpstatus!~ /offset/) {&bye („Kann Zeitserver nicht
erreichen!“,0);}
```

Hierbei wird die Variable $ntpstatus mit dem Ergebnis des ntpdate-Befehls geladen. Die angegebenen Server sind verschiedene, frei zugängliche Zeitserver in Deutschland. Die örtliche Nähe soll dabei die Laufzeit der Pakete klein halten und Probleme mit verschiedenen Zeitzonen sowie Sommer- und Winterzeit vermeiden. In der nächsten Zeile wird überprüft, ob das Ergebnis den Wert offset enthält – er beschreibt die Diskrepanz zwischen lokaler und Remote-Zeit. Kommt offset nicht vor, so konnte der Zeitserver offenbar nicht erreicht werden und das Programm bricht ab.

4.11.5. Synchronisation

Im Hauptteil des Updater-Skriptes werden nun die zur Synchronisation benötigten Routinen durchgeführt. Dabei handelt es sich im Einzelnen um (in Reihenfolge der Ausführung):

- Logtransfer: Als erste Aktion werden die lokal gespeicherten Log-Einträge auf den Server übertragen. Falls das Skript im späteren Verlauf durch einen Fehler abbricht, so wird die aufgetretene Fehlermeldung beim nächsten Durchlauf an den Server übermittelt und ist somit auf der Webseite einsehbar.
- Format-Update: Es ist möglich, für jeden angeschlossenen Client ein eigenes Format zu definieren. In diesem Abschnitt wird die config Datenbank des

Server gelesen und der Eintrag für die gewählte Client Nummer ($client) in die lokale Datenbank geschrieben.

- Löschen von Titeln: Wie bereits in Abs. 4.11.2 erläutert ist es sinnvoll, zuerst die gelöschten Titel aus der Datenbank zu entfernen. Dazu liest das Skript den Zeitpunkt des letzten Updates aus der lokalen Tabelle config ein und speichert diesen in $lastupdate. Daraufhin werden die gelöschten Titel mittels folgender Abfrage ermittelt:

```
SELECT filename FROM deleted
                WHERE deleted > $lastupdate';
```

Dabei zeigt *MySQL* ein hohes Maß an Flexibilität, da es sich beim Vergleich deleted > ,$lastupdate' nicht um rein numerische Werte (das Format ist zum Beispiel 2004-02-02 13:21) handelt. Im Weiteren werden nun durch die Anweisung

```
DELETE FROM archive WHERE filename='$filename'
```

die Einträge mit entsprechenden Dateinamen aus der lokalen Datenbank getilgt. Dieser Vorgang wird anschließend für die Jingles, Beiträge und Werbung wiederholt. Um welche Art von Programmelement es sich handelt erkennt das Skript anhand der Spalte Type in der Tabelle deleted.

- Download der neuen Titel: In diesem Programmabschnitt werden Musikstücke und Programmelemente mittels der FTP-Schnittstelle auf den Client übertragen. Dabei kommt wie schon zuvor der Timestamp des Uploads für die Identifikation neuer Titel zum Einsatz. Wir werden im folgenden Kapitel genauer auf die Besonderheiten des skriptgesteuerten Dateitransfers eingehen, da es hier vor allem wegen der häufigen Verbindungsabbrüche zum Server die meisten Schwierigkeiten gab.
- Aktualisierung der Elemente: Wurden Elemente auf dem Server verändert, so ist dies in der Tabelle updated dokumentiert. Dort findet sich eine

Referenz auf den Zähler (counter) des aktualisierten Titels. Durch die Abfrage

```
SELECT updateid,type FROM updated WHERE updated >
'$lastupdate'
```

wird dieser Zähler sowie die Art des Programmelements ermittelt. Daraufhin wird der entsprechende Datensatz vom Server eingelesen und per

```
UPDATE $whichtable SET
artist=$artist,songtitle=$songtitle,
            play_limit=$play_limit
                 WHERE filename=$filename"
```

in die lokale Datenbank geschrieben.

- Aktualisierung der Playlist: Sind alle neuen Titel vom Server übertragen, so wird die vom Anwender eingestellte Playlist übermittelt. Als erster Schritt wird dabei die Anzahl der Titel pro Durchlauf (welche in der Tabelle planned_songs gespeichert ist) eingelesen und lokal entsprechend angepasst. Im weiteren Verlauf wird die Tabelle planned_songs_list zeilenweise eingelesen und ebenfalls lokal gespeichert. Hier verbergen sich allerdings wiederum einige Fallen bei der Synchronisation, daher werden wir den Ablauf im folgenden Abschnitt genauer behandeln.
- Statistische Auswertung: Die Client speichert die Anzahl der Wiederholungen einzelner Programmelemente (nicht von Musikstücken) und überträgt diesen Wert beim Update an den Server. Zu diesem Zweck werden zwei Queries durchgeführt:
    - Tabelle statistics: Für die statistische Auswertung wird der Eintrag des entsprechenden Elements in der Tabelle statistics erhöht.
    - Archiv: Um zu bestimmen, wie häufig ein Titel bereits gespielt

wurde, nimmt die Spalte played in den entsprechenden Archiv-Tabellen (ad_archive, feature_archive bzw. jingles_archive) die Anzahl der Wiederholungen auf.

Sind die Daten übertragen, wird die lokale statistics-Tabelle des Clients geleert, damit diese über einen längeren Zeitraum nicht zu groß wird.

- Aktualisierung lastupdate: Abschließend aktualisiert das Skript den Zeitpunkt des letzten Updates für den Client, sowohl in der lokalen als auch entfernten config Tabelle. Dabei ist zu beachten, dass dieser Zeitpunkt nicht die Uhrzeit beim Beenden des Skriptes, sondern bei dessen Start ist. Dies dient dazu, Veränderungen auf dem Server, welche während des Update-Vorganges getätigt wurden, beim nächsten Durchlauf der Software nachzuholen.

4.11.6. Dateitransfer

Um neue Lieder und Elemente auf den Client zu übertragen, verwendet das DIRA-System das FTP-Protokoll. Dessen Nutzung wird in *Perl* über das Net::FTP Modul gesteuert, welches seit Version 5.6 zum Standardumfang von *Perl* gehört. Die Bedienung erfolgt über Handles, diesmal allerdings so genannte Transfer-Handles.

Der Befehl

```
$ftp = Net::FTP->new(„www.tektribe.de“,Timeout =>60)
```

definiert das Transfer-Handle und baut eine Verbindung zum angegebenen FTP-Server auf. Über eine Reihe von weiteren Anweisungen lässt sich nun die Übertragung kontrollieren. Zuerst muss sich der Client am Server autorisieren, was mittels

```
$ftp->login($benutzer,$ftppw)
```

geschieht. Die Variablen $benutzer und $ftppw wurden zu Beginn des Skriptes initialisiert. Daraufhin kann mit Hilfe von

```
$ftp->get($ftpfilename,$filename)
```

eine Datei vom Server auf den Client übertragen werden. Bei $ftpfilename handelt es sich um den Dateinamen ohne vorangestelltes /home, da dieses wegen des geänderten Wurzelverzeichnisses (chroot) auf dem Server wegfällt. Weitere benutzte Befehle sind $ftp->binary, was die Übertragung von Binardateien aktiviert sowie $ftp->size($ftpfilename), welches die Dateigröße auf dem Server zurückliefert.

Wie bereits erwähnt, gab es bei der Übertragung der Dateien einige Schwierigkeiten, welche sowohl auf die schlechte Anbindung des Servers als auch auf die Besonderheiten der Timestamp-Synchronisation zurückzuführen sind.

Um die Stabilität der Software zu überprüfen, wurden 20 neue Musikstücke auf den Server übertragen. Da der Transfer zum Client über eine ISDN-Anbindung erfolgt, dauert dieser bei einer durchschnittlichen Dateigröße von 3-4 Megabyte etwa 3 Stunden. Während des FTP-Transfers kam es häufig zu Verbindungsabbrüchen, was dazu führte, dass das Skript über die &bye Routine abbrach und den Zeitpunkt des letzten Updates nicht aktualisierte. Beim nächsten Versuch wurden dann alle bereits übertragenen Titel erneut angefordert. Um das zu verhindern, wurde das Skript dahingehend erweitert, dass vor der Übertragung die Größe der lokalen und der entfernten Datei ermittelt und überprüft wird. Sind beide Werte identisch, so wird von einer erneuten Übertragung abgesehen.

Aber auch diese Modifikation führte nicht zum gewünschten Ergebnis, da bei einem abgebrochenen Transfer das Skript manchmal nicht ordnungsgemäß über die &bye Routine abbrach, sondern mit einer Fehlermeldung im Net::FTP Modul starb. Das führte dazu, dass die Lock-Datei nicht gelöscht wurde und somit keine weiteren

Updates möglich waren. Über die Ursachen dieses Fehlers kann man nur spekulieren, es handelt sich wohl um einen Softwarefehler im FTP-Modul von *Perl*. Nach einigem Probieren ließ sich das Problem über eine Hilfskonstruktion mit dem eval{} Befehl lösen. Dieser führt die Befehle in der geschweiften Klammer quasi überwacht aus und schreibt etwaige Fehlermeldungen in die Variable $@. Im Quelltext findet man daher folgende Anweisungen:

```
eval {$ftp->get($ftpfilename,$filename)};
      if ($@) {&bye („Download von Datei fehlgeschlagen,
neuer Versuch beim naechsten Update, $!“,0);}
```

Dadurch konnten die Verbindungsabbrüche abgefangen werden.

4.11.7. Synchronisation der Playlist

Da der Anwender auf der Webseite die Möglichkeit hat, eine individuelle Playlist zu erstellen, die sich auf den Datenbestand des Servers bezieht, muss man bei der Übertragung dieser auf den Client die Einträge dem Datenbestand des Clients anpassen. Dies ist nötig, wenn ein bestimmtes Programmelement fest auf einem Platz in der Playlist eingeplant ist.

Da in diesem Fall in der Spalte specific der Tabelle planned_songs_list lediglich ein Verweis auf einen Zähler im Archiv gespeichert ist, muss der Client das gewünschte Element aus dem Archiv des Server einlesen. Daraufhin sucht er dieses in der lokalen Datenbank und übernimmt dessen Zähler für seine eigene Playlist. Die Tatsache, dass der gleiche Titel auf Client und Server eine unterschiedliche Id haben können, liegt an der verteilten Struktur des Systems. Werden etwa drei neue Titel auf den Server hochgeladen (z.B. mit den Id’s 78, 79, 80) und einer davon vor dem Transfer zum Client gelöscht, so hätte der nächste Eintrag auf dem Server die Id 81 und auf dem Client die 80.

Ein weiteres Problem ist, dass es für Veränderungen in der Playlist keine

Timestamps gibt. Dies wird an folgendem Beispiel deutlich:

Ein Client führt um 12 Uhr ein Update durch, welches eine halbe Stunde in Anspruch nimmt, da einige neue Musikstücke übertragen werden müssen. Während dieser Zeit (etwa um 12:05) lädt der Benutzer ein neues Jingle auf den Server und plant dieses fest auf eine Position in der Playlist ein. Der Client wird dieses Jingle nicht mehr übertragen, da der Jingle-Abgleich bereits abgeschlossen ist. Bei der Synchronisation der Playlist (welche nicht durch Timestamps gesteuert wird) kann er nun das gewünschte Element nicht in seiner lokalen Datenbank finden.

Für diesen Fall findet sich im Skript eine Verzweigung, welche diesen Fehler abfängt und stattdessen den Wert NULL an die entsprechende Position der Playlist schreibt. Dies entspricht der Einstellung „Spiele ein beliebiges Element“. Beim nächsten Durchlauf des Updates wird der neue Jingle dann übertragen und auch entsprechend in die Playlist eingebaut.

Ursprünglich war geplant, in diesem speziellen Fall die Synchronisations-Routinen für das entsprechende Archiv (also jingles_archive) erneut durchzuführen. Dies hätte den Code aber sehr unübersichtlich gemacht, da ja während der Neu-Übertragung ein weiteres Element durch den Benutzer hinzugefügt werden kann, was wiederum hätte abgefragt werden müssen. Da der Zeitpunkt des letzten Updates in diesem Beispiel nach Beenden des Skriptes auf 12:00 Uhr gesetzt wird, erscheint es außerdem logisch, dass eine Veränderung, welche erst um 12:05 getätigt wurde, nicht mehr durchgeführt wird[131].

Hat das Updater-Modul die beschriebenen Aufgaben durchgeführt, so verfügt der Client über einen zum Server synchronen Datensatz. Dies stellt eine Bestätigung von These 2 dar.

[131] [INW1994], S. 119 - 127

4.12. Das Feedme-Modul

Dieses Modul dient dazu, die zu spielenden Titel an die *Jukepeg*-Software zu übergeben. Es ist komplett gekapselt, das bedeutet, dass es nicht weiß, was das Update-Skript gerade macht. Es verlässt sich ausschließlich auf die Informationen in der lokalen Datenbank. Es handelt sich bei dieser Software um eine Weiterentwicklung des *tox.pm* Moduls. Die Kommunikation findet dabei aber nicht über die von *ices*[132] (einer MP3-Streaming Software) vorgegebenen Subroutinen statt, sondern über Sockets.

Bei Sockets handelt es sich um bidirektional nutzbare Schnittstelle zur Kommunikation zwischen Programmen. Im Falle des DIRA-Systems stellt die *Jukepeg*-Software ein Socket Jukepeg.sock im Verzeichnis /tmp zur Verfügung, über welches sie sich steuern lässt. Daher werden zu Beginn die Konstanten für diese Schnittstelle definiert und eine Verbindung aufgebaut:

```
sub SOCKNAME        { ,/tmp/jukepeg.sock'; }
sub SOCKHOST        { ,localhost'; }
$socket = IO::Socket::UNIX->new(Type => SOCK_STREAM,
Peer => SOCKNAME);
```

Damit dieser Aufruf funktioniert, muss vorher mittels `use IO::Sockets` das entsprechende *Perl*-Modul geladen werden. Im Folgenden kann nun über die Variable $socket die Wiedergabe über *Jukepeg* gesteuert werden. Zu diesem Zweck werden Befehle in Textform in das Socket geschrieben und die entsprechenden Antworten gelesen und verarbeitet. Die Kommunikation zu Beginn der Wiedergabe würde sich z.B. folgendermaßen darstellen:

[132] http://www.icecast.org

→ select 0
← 250
→ flush
← 250
→ queue /home/dira/winston-wovonlebteigentlichpetermp3
← 251

Tabelle 4.12a – Beispiel für die Socket Kommunikation

Die mit → bezeichneten Befehle sind Daten, die in das Socket geschrieben werden, Zeilen mit ← werden daraus gelesen. Um den Code besser lesbar zu machen sind die numerischen Antworten von *Jukepeg* als Konstanten definiert. Dabei handelt es sich um:

Wert	Bedeutung	Konstante
250	Operation erfolgreich	RESP_OKAY
251	Titel erfolgreich in Warteschlange aufgenommen	RESP_QUEUED
252	Titel in Warteschlange erfolgreich wiedergegeben	RESP_FINISHED
434	Warteschlange voll	RESP_FULL
412	Soundkarte belegt	RESP_IN_USE

Tabelle 4.12b – Liste der Antwortcodes

Die Verwendung dieser Antwortcodes wird in folgendem Abschnitt behandelt.

4.12.1. Benutzte Subroutinen

Analog zu Abs. 4.11.3 sollen hier zuerst die im *Feedme.pl* Skript benutzten Subroutinen behandelt werden. Dabei unterscheiden wir zwischen socket- und datenbankbezogenen Routinen:

Socketbezogen:

- check_response(ANTWORT): Diese Subroutine wartet so lange, bis sie die in ANTWORT festgelegte Ausgabe über das Socket erhält. Das geschieht über folgenden Code:

```
while(<$socket>) {
        […]
if(m/^(\d{3}) .*$/) {
return $1 if(!defined($_[0]) || $1 == $_[0]);
[…]
die "$Skript: unexpected response - got $1,
expecting $_[0]\n" if $1 != RESP_FINISHED;
                    }
                  }
```

  Dabei werden so lange Daten vom Socket gelesen, bis das Pattern Matching (if(m/^(d{3} …))) eine dreistellige numerische Zahl erhalten hat. Diese wird der Variablen $1 zugewiesen. Entspricht diese dem Wert von ANTWORT (dargestellt als $_[0] – dem ersten an die Subroutine übergebenen Wert), wird sie zurückgegeben. Bei allen weiteren Werten (außer RESP_FINISHED) bricht das Skript die Bearbeitung ab. Die RESP_FINISHED Ausnahme ist deswegen notwendig, da es passieren kann, dass ein Titel genau in dem Moment fertig abgespielt wurde, in dem ein Befehl abgesetzt wird. Das führt dazu, dass der Rückgabewert nicht RESP_OK sondern eben RESP_FINISHED ist. In diesem Fall soll das Skript aber nicht abbrechen, sondern weiter auf die „richtige“ Antwort warten.

- write_command(BEFEHL, ANTWORT): Diese Subroutine erwartet zwei oder einen Parameter: Zum einen den an das Socket zu übertragenden Befehl (dieser Wert muss angegeben werden), zum anderen den erwarteten Rückgabewert. Die Ausführung benötigt nur 3 Zeilen Code, da sie auf eben

beschriebene Subroutine check_response zurückgreift.

```
Sub write_command
{(@_ && @_ < 3) or die;
     die "$Skript: failed to write command\n"
unless print $socket "$_[0]\n";
return &check_response($_[1]);
}
```

In der ersten Zeile wird überprüft, ob die richtige Anzahl an Parametern übergeben wurde (`@_ < 3`), bzw. ob überhaupt Parameter vorhanden sind (`@_`). Daraufhin wird mittels `print $socket „$_[0]\n“` der erste Parameter (also BEFEHL) an das Socket geschrieben. Schlägt dies fehl, so bricht das Skript ab. Abschließend wartet das Skript auf ANTWORT – dies wird mit Hilfe der check_response Routine bewerkstelligt. Ist ANTWORT nicht definiert, so wird auf Grund der `(!defined($_[0])` Anweisung in der check_respone Subroutine der zurückgegebene Wert ausgegeben.

- queue_fill(DATEINAME, GENRE): Diese Routine benötigt beide Parameter. Die Angabe des Genres ist notwendig, da die Überblendzeiten zwischen zwei Titeln für Musikstücke und Programmelemente unterschiedlich ausfallen sollen. So ist etwa bei Wortbeiträgen ein Überblenden nicht erwünscht während dies bei Musikstücken zu einem besseren Programmfluss führt. Bei DATEINAME und GENRE handelt es sich, anders als bei den bisherigen Routinen, nicht um eine einfache Variable, sondern um Arrays – man kann sie sich als Listen von etwa Titeln vorstellen. Diese Listen werden von der create_queue Subroutine erzeugt. Bei der Abarbeitung wird nun jeweils das erste Element der Listen ($queue[0] bzw. $style[0]) ausgelesen und folgende Befehle in Abhängigkeit von GENRE ausgeführt:

```
if ($style[0] eq ‘Music’)
```

```
{&write_command('fadeout 15 ', RESP_OKAY);
 &write_command('fadein 15', RESP_OKAY);
 &write_command('overlap 30', RESP_OKAY);
} elsif ($style[0] eq 'Jingles')
{&write_command('fadeout 0', RESP_OKAY);
 &write_command('fadein 0', RESP_OKAY);
 &write_command('overlap 0', RESP_OKAY);
[…]
$resp = write_command('queue $queue[0]') ;
```

Handelt es sich bei $style[0] um Musik, so wird mittels der fadeout-, fadein- und overlap-Befehle die Überblendzeit auf 1,5 Sekunden eingestellt. Bei etwa einem Jingle ist dieser Wert 0 – es wird keine Kreuzblende durchgeführt. Abschließend wird der in $queue[0] gespeicherte Dateiname in die Warteschlange von *Jukepeg* geschrieben. Ist dies geschehen, so wird das erste Element von $style bzw. $queue mittels des shift-Befehls gelöscht.

Diese Vorgänge werden nun so lange wiederholt, bis beide Arrays keine weiteren Elemente mehr enthalten.

Datenbankbezogen:

- notplayed(COUNTER): Mit dieser Routine wird überprüft, ob ein Titel bereits gespielt wurde. Zu diesem Zweck wird COUNTER mit sämtlichen Elementen des lastplayed Arrays verglichen. Werden keine Treffer gefunden, so ist der Rückgabewert 1, andernfalls 0.
- select_queue: Das *Feedme.pl* Modul benutzt aus Kompatibilitätsgründen verschiedene Warteschlangen. Die select_queue Subroutine behandelt dabei die in der *MySQL*-Datenbank gespeicherten Werte. Die Funktion dieser Routine ist simpel: Sie liest die ersten Werte der Warteschlange aus, löscht diesen und schiebt die folgenden Positionen jeweils einen Wert nach oben (d.h. Position 2 wird zu 1, 3 zu 2, usw.). Daraufhin werden die Id sowie die Art des Programmelements (Musik, Jingle, Beitrag oder Werbung)

zurückgegeben.

- jingle, feature, ad (SPECIFIC, ELEMENTE): Hierbei handelt es sich um Routinen zur Auswahl von Programmelementen. Wird der Parameter SPECIFIC übergeben, so wird das explizit geplante Element der Playlist ausgewählt. Ist dies nicht der Fall, so geschieht die Auswahl, anders als bei den Musikstücken, nach der Anzahl der Wiederholungen einer Datei – so wird zum Beispiel der am wenigsten gespielte Beitrag ausgesucht. Sind mehrere Beiträge geplant, so wird durch die Übergabe von ELEMENTE dafür gesorgt, dass etwa der zweite geplante Beitrag das am zweit-seltensten gespielte Element ist.
- select_rand: Tritt bei der Abarbeitung der Playlist ein Fehler auf, so wird mit dieser Subroutine ein zufälliger Titel ausgesucht. Dies dient der Erhöhung der Betriebsstabilität.

4.12.2. Erstellen der Playlist

Die Aufgabe des *Feedme.pl* Skripts darin, eine Playlist anhand der in der Datenbank gespeicherten Vorgaben zu erstellen. Dabei kommen insgesamt drei verschiedene Warteschlangen (Queues) zum Einsatz. Dies wird durch folgende Grafik deutlich:

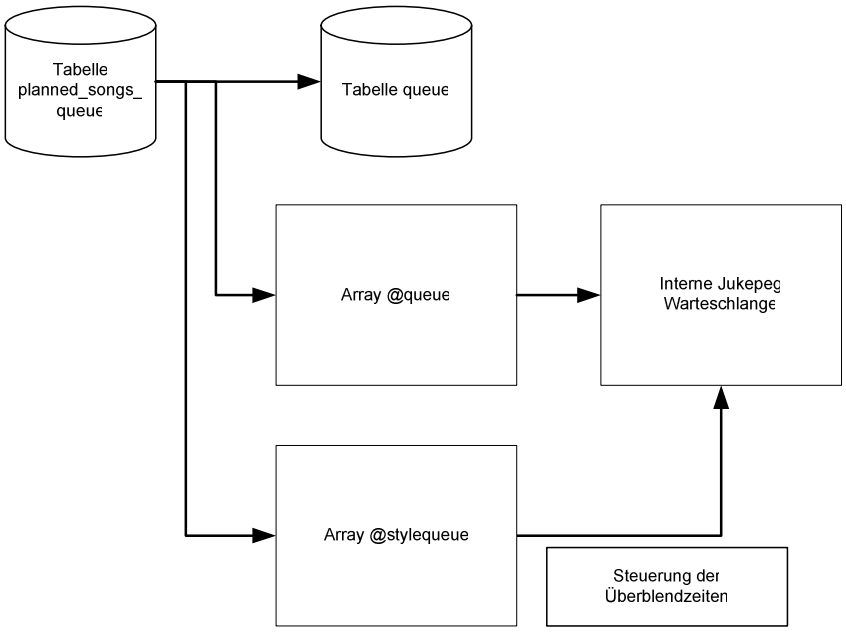

Abbildung 4.12.2a – Schema der Warteschlangen

Die Vorgaben sind dabei in der Tabelle planned_songs_queue gespeichert. Wird eine neue Playlist erstellt, so werden die ausgewählten Titel und Meta-Informationen in folgenden Datenstrukturen gespeichert:

- Tabelle queue: Um auf den gerade gespielten Titel und die noch zu spielenden Elemente leicht über die *MySQL*-Schnittstelle zugreifen zu können, werden die Daten in der Tabelle queue gespeichert. Dies ist für zukünftige Funktionen der Software vorgesehen und hat im Moment noch keine Funktion.
- Array queue: In diesem Array werden die Dateinamen der zu spielenden Titel abgelegt. Dabei entspricht Position 0 Element 1, Position 1 Element 2 usw.
- Array style: Um die Übergangszeiten zwischen Musiktiteln und Programmelementen steuern zu können, wird in diesem Array gespeichert, von welchem Typ das entsprechende Element ist.
- Interne *Jukepeg* Warteschlange: Damit *Jukepeg*-Elemente überblenden kann, muss die Software bereits vor dem Ende eines Titels wissen, welche Datei

als nächstes zu spielen ist. Daher wird über das Socket eine Liste von Dateien sowie die dazugehörigen Überblendzeiten übergeben.

Die interne Warteschlange wird über die in vorherigem Abschnitt beschriebene &queue_fill Subroutine abgearbeitet, alle anderen über die &create_queue Subroutine.

Zu Beginn dieser Routine wird aus der Tabelle planned_songs die Anzahl der zu spielenden Titel ausgelesen und in der Variable $numberofsongs gespeichert. Daraufhin werden aus der Tabelle planned_songs_list die Werte type, specific und format für die entsprechende Position der Warteschlange ausgelesen. Dies geschieht von 1 bis zu dem in $numberofsongs festgelegten Wert.

Der Wert von type bestimmt dabei die Art des Programmelements. Entspricht er zum Beispiel „music“, so wird ein Titel aus dem Musikarchiv ausgewählt. Die Spalte specific ist nur für Jingles, Beiträge oder Werbung relevant: Sie gibt die Id des fest vorgeplanten Elements an. Ist der Wert NULL, so wird ein beliebiges Element ausgewählt. Der Inhalt von format dagegen gilt nur für Musikstücke – er gibt das zu spielende Genre an.

In folgendem Beispiel soll der Ablauf der Subroutine für ein Jingle-Element beschrieben werden:

```
[…]
elsif ($type eq 'Jingles')  {
            $id = &jingle($specific,$num_jingles);
[…]
            $num_jingles++;
```

In diesem Abschnitt wird durch den Aufruf der &jingles Routine die Id des zu spielenden Jingles zurückgegeben. Ist ein spezielles Element geplant, so entspricht $specific gleich $id. $num_jingles bezeichnet, um das wievielte Jingle-Element es sich in dieser Playlist handelt.

```
$query="SELECT filename FROM jingles_archive WHERE
counter = '$id'";
[…]
$tempfile2 = '/tmp/queuedsong'.$tempvar.'.mp3';
system ("cp $tempfile1 $tempfile2");
```

Um zu verhindern, dass das *Updater.pl*-Skript einen Titel löscht, der sich bereits in der Warteschlange des Players befindet, werden alle zu spielenden Titel als temporäre Dateien angelegt. Zu diesem Zweck wird mittels des system() Befehls eine Kopie des durch die Datenbankabfrage ermittelten Dateinamen angelegt.

```
push @queue,$tempfile2;
push @stylequeue,Jingles;
++$tempvar;
```

Abschließend wird nun der so erzeugte Dateiname in das @queue Array geschrieben, das @stylequeue Array erhält den Wert Jingles.

Die Verarbeitung von anderen Elementen wie Musiktitel oder etwa Werbeblöcken geschieht analog zum eben beschriebenen Programmausschnitt. Diese Verarbeitung wird so lange durchgeführt, bis der in $numofsongs festgelegte Wert erreicht ist.

Ist die &create_queue Subroutine abgeschlossen, so stehen dem Skript nun zwei Arrays zur Verfügung. Dabei handelt es sich zum einen um eine Liste von Dateinamen (@queue), etwa in der Form „/home/ads/werbung1mp3;/home/music/chemcialbrothers.mp3;…“, zum anderen um eine Liste mit den entsprechenden Programmelementen (z.B. „ads“,“music“,…).

4.12.3. Wiedergabe der Playlist

Da beim Abspielen der Titel auf dem Client eine kontinuierliche Wiedergabe gewünscht wird, wird dieser Abschnitt in einer Endlosschleife wiederholt. Zuvor

werden durch die Anweisungen

```
&write_command('flush', RESP_OKAY);
@queue=();
@stylequeue=();
$query="DELETE FROM queue";
```

die interne Warteschlange von *Jukepeg*, die beiden Warteschlangen-Arrays sowie die in der Tabelle queue gespeicherten Titel gelöscht. In der darauf folgenden Endlosschleife wird überprüft, ob die Warteschlangen leer sind, was beim ersten Durchlauf immer zutreffend ist.

```
for (;;) {
[…]
$query="SELECT id FROM queue ORDER BY position";
$sth=$dbh->prepare($query);
$sth->execute();
$entries = $sth->rows;
$sth->finish();

if ($entries == 0) {
          &create_queue();
          &queue_fill (\@queue,\@stylequeue);
                    }
```

Durch die im vorherigen Kapitel beschriebenen Routinen &create_queue und &queue_fill wird eine neue Playlist erstellt und *Jukepeg* erhält die in den Arrays @queue und @stylequeue gespeicherten Daten. Die Arrays sind nach dieser Aktion leer und werden bis zum nächsten Erstellen der Playlist nicht mehr benutzt.

Im nächsten Abschnitt liest das *Feedme.pl* Skript aus der Tabelle queue der Datenbank den nächsten zu spielenden Titel aus – in unserem Beispiel also Titel 1. *Jukepeg* hat in der Zwischenzeit bereits mit der Wiedergabe dieses Titels begonnen.

Abhängig davon, um welche Art von Programmelement es sich handelt, werden nun verschiedene Aktionen durchgeführt:

- Musikstücke: Ist der gerade gespielte Titel ein Lied, so wird die Anzahl der Wiederholungen in der Tabelle archive erhöht.
- Andere Programmelemente: Auch hier wird die Anzahl der Wiederholungen erhöht. Zusätzlich findet noch ein Eintrag in die Tabelle statistics statt, mit dessen Hilfe das Updater-Skript die Statistikdaten des Servers aktualisieren kann.

Unabhängig von der Art des Elements befindet sich am Ende jedes Blocks folgende Anweisung:

```
&check_response(RESP_FINISHED);
```

Damit wird das Skript angewiesen, so lange zu warten, bis es von *Jukepeg* die Meldung erhalten hat, dass der aktuelle Titel fertig gespielt wurde.

An dieser Stelle soll erklärt werden, warum das recht komplizierte Konstrukt aus zwei Arrays und der queue-Tabelle notwendig ist:

Damit *Jukepeg* zwischen den Titeln überblenden kann, muss es vor dem Ende eines Elements den Dateinamen des nächsten Titels wissen. Da die Steuerung des Programms aber weiterhin durch die Daten in der *MySQL*-Datenbank geschehen sollte, war die Lösung mit den Arrays notwendig. Bevor die Tabelle queue Eintrag für Eintrag durchgegangen wird, hat *Jukepeg* bereits alle Titel in der internen Warteschlange gespeichert. Durch das Warten auf RESP_FINISHED wird nun sichergestellt, dass sich beide Programmteile einig sind, an welcher Stelle der Playlist sie sich befinden.

Ist kein weiteres Element mehr in der Tabelle queue enthalten, kehrt das Programm

an den Anfang der Endlosschleife zurück und erstellt eine neue Playlist mittels der &create_queue Routinen.

4.13. Simulation

Bevor das System in der Praxis erprobt wird, ist ein Simulationslauf erforderlich. Die dadurch gewonnen Daten dienen zur Überprüfung von These 3,4,7 und 8.

Die hier gewonnenen Zahlen bezüglich der Anzahl der möglichen Benutzer haben nur eingeschränkte Aussagefähigkeit auf die tatsächliche Leistungsfähigkeit des Systems, da Client und Server nicht sehr leistungsstark sind. Es soll allerdings versucht werden, ungefähre Werte für professionelle Serverhardware zu extrapolieren.

4.13.1. Leistungsfähigkeit des Clients

Auf der Client-Seite sind die Anforderungen an die Hardware äußerst gering, wenn auch nicht so niedrig wie bei einem Streaming-System. Durch Erhöhung des durch die Onboard-Grafikkarte genutzten Arbeitsspeichers konnte das Verhalten des Systems bei noch kleinerem Hauptspeicher getestet werden. Da die Grafikkarte über keinen eigenen Speicher verfügt, reserviert sie Teile des Arbeitsspeichers, der dann dem System nicht mehr zu Verfügung steht.

Bei der Entwicklung des Systems waren dabei 8 MByte für die Grafikkarte reserviert, die restlichen 120 MByte standen für das Betriebssystem zur Verfügung.

Dabei zeigte sich folgende Speicherauslastung durch die verwendete Software:

Programm	Instanzen	Speicher/Instanz	Speicher gesamt
Mysql-Server	3	10,8 MByte	32,5 MByte
Feedme-Modul	1	6,1 MByte	6,1 MByte
Updater-Modul	1	< 10 MByte	< 10 MByte
Jukepeg	2	2,8 MByte	5,6 MByte
		Benutzter Speicher	Ca. 54,2 MByte

Tabelle 4.13.1a – Serverlast durch die verschiedenen Module

Dabei ist anzumerken, dass der verwendete Speicher des Updater-Moduls stark schwankt. Dies ist wohl auf interne Caching-Mechanismen von *Perl* bei den Datenbankanfragen und während des Dateitransfers zurückzuführen.

Mit dieser Konfiguration muss das System keinen Arbeitsspeicher auf die Festplatte auslagern, d.h. es konnten alle Daten im Hauptspeicher gehalten werden. Der restliche Speicher wird durch den Kernel, die Treiber sowie die Shell belegt. Es sei erwähnt, dass keine grafische Oberfläche eingesetzt wurde.

Erhöht man den Grafikkartenspeicher auf 32 MByte und verringert somit den Hauptspeicher auf 96 MByte, so steigt die Menge des ausgelagerten Speichers auf etwa 22 MByte an. Da die Zugriffsgeschwindigkeit auf diesen ausgelagerten Speicher natürlich wesentlich geringer als auf herkömmlichen Speicher ist, könnten dadurch Verzögerungen bei der Wiedergabe oder den Datenbankanfragen auftreten. In einem mehrstündigen Test waren aber keine Probleme feststellbar.

Stellt man dem Betriebssystem nur noch 64 MByte zur Verfügung, so wird das System sehr langsam. Zwar gelingen das Booten und die Wiedergabe der MP3-Dateien noch weitgehend störungsfrei, da sich aber scheinbar die komplette Datenbank im Auslagerungsspeicher befindet, dauern Anfragen teilweise so lange, dass Pausen von bis zu mehreren Sekunden zwischen den Titeln auftreten.

Da aber mittlerweile keine Speichermodule mit einer geringeren Kapazität als 128 MByte im Handel erhältlich sind, ist diese Einschränkung für den Betrieb mit Standard-PC Hardware akzeptabel.

Die Rechenleistung des Client-Rechners ist dabei völlig ausreichend, die größte Auslastung der CPU tritt dabei während der Updates auf, diese überschreitet (wohl auch wegen der durch das Internet verringerten Übertragungskapazität) nie 5 % der Gesamtrechenleistung. Die beiden Instanzen von *Jukepeg* nehmen dabei außerdem kontinuierlich etwa 3 % der Leistung in Anspruch. Bei einer Frequenz von 24 Updates pro Tag liegt die Auslastung bei durchschnittlich 2 %.

Auf Grund der geringen Anforderung wäre es wohl möglich, den Client in einem Embedded System zu integrieren. Diese oft nur wenige Zentimeter großen Rechner sind mittlerweile mit Netzwerkanschluss, Audioausgang und interner Festplatte verfügbar. Als Betriebssystem kommt dabei eine auf geringe Speicherkapazität und Rechenleistung angepasste Linux-Variante zum Einsatz. Leider sind diese Systeme noch wesentlich teurer als normale PC-Hardware.

4.13.2. Leistungsfähigkeit des Servers

Die vom Server benötigte Rechenleistung für die Bedienung eines einzelnen Clients ist auf Grund der verteilten Systemarchitektur geringer als beim TOX-System. Da das System für eine große Anzahl von Clients eingesetzt werden kann, soll im Folgenden die Leistungskapazität des verwendeten Servers abgeschätzt werden. Anhand dieser Daten kann eine Interpolation auf aktuellere Serverhardware vorgenommen werden.

Bei Messungen der Systemlast für die Transfers wurde eine 20 MByte große Datei übertragen, die Auswertung erfolgte mit dem Programm *top*. Folgende Werte konnten ermittelt werden:

Aktion	Geschwindigkeit	Speicher-Nutzung	CPU-Last
Download von Dateien per DSL	Ca. 200 kByte/s	0,7 %	0,6 %
Download von Dateien per ISDN	7 kByte/s	0,7 %	0,3 %

Tabelle 4.13.2a – Last bei Dateitransfers

Hier ist ersichtlich, dass der Transfer kaum Ressourcen bindet.

Für die Auswertung der durch die *MySQL*-Datenbank erzeugten Last musste zuerst die Art der Anfragen aufgeschlüsselt werden. Nach 7 Monaten Betrieb zeigte sich dabei unter insgesamt 192.981 Anfragen folgende Verteilung:

Aktion	Anzahl	Anteil aller Anfragen
Neue Datensätze (INSERT)	16.833	8,72 %
Lesen von Datensätzen (SELECT)	134.399	69,64 %
Aktualisieren von Datensätzen (UPDATE)	16.088	8,34 %
Löschen von Datensätze (DELETE)	1.926	1 %

Tabelle 4.13.2b – Verteilung der MySQL-Anfragen, fehlende Werte zu 100 % bezeichnen weitere, nicht genannte Anfragen

Diese Werte beinhalten sowohl die Anfragen durch den Webserver als auch durch die Clients. Um die Einwirkung des Webservers sowie von *PHP* auf die Gesamtperformance abschätzen zu können, wurde ein *PHP*-Skript mit folgender Funktionalität erstellt:

- Aufbau der Datenbankverbindung
- Auswahl von 7.000 zufälligen Einträgen
- Einfügen von 1250 weiteren Elementen
- Aktualisierung von 1250 zufälligen Einträgen
- Löschen von 100 zufällig ausgewählten Reihen

Als Testdatenbank kam dabei eine separate Datenbank mit einer Tabelle zum Einsatz. Der Aufbau der Tabelle entspricht der Tabelle archive des DIRA-Systems, per Skript wurden 10.000 Zeilen mit Zufallswerten in diese als Ausgangswerte eingetragen. Dieses *PHP* Skript wurde mittels des *Apache*-Benchmark Tools *ab*[133] aufgerufen. Um die Last auf den Server zu erhöhen, steigert man die Anzahl der gleichzeitigen Anfragen.

Anzahl gleichzeitige Anfragen	Mysql	Apache / PHP	Last Benutzer-prozesse	Last System-prozesse	Gesamt-last
1	14,1 %	15,3 %	30,3 %	7,9 %	38,2 %
2	20,3 %	27,2 %	47,5 %	16,3 %	63,8 %
3	26,7 %	36,7 %	63,4 %	22,7 %	86,1 %
4	36,5 %	36,9 %	73,4 %	24,5 %	97,9 %
5	37,2 %	38,2 %	75,4 %	25,5 %	99,9 %

Tabelle 4.13.2c – Last des Servers

Aus dieser Untersuchung lassen sich folgende Schlüsse ziehen:

- Bereits bei 2 gleichzeitigen Anfragen steigt die Last durch die Systemprozesse um mehr als das Doppelte. Dies ist wohl damit begründet, dass nicht genügend Arbeitsspeicher zur Verfügung steht und Tabellen

[133] http://httpd.apache.org/docs/programs/ab.html

ausgelagert werden müssen. Dadurch entsteht eine Belastung des I/O Systems (Festplatte, Controller usw.), welche die Systemlast erhöht.

- Da *MySQL* in der vorliegenden Konfiguration maximal 3 Instanzen startet, erhöht sich bei 4 gleichzeitigen Anfragen die Last durch den *MySQL*-Server sprunghaft. Dies liegt an daran, dass nun eine laufende Instanz zwei verschiedene *Apache*-Prozesse bedienen muss, was sich in einer Verschlechterung des Queuings und somit einer Erhöhung der Prozesslast bemerkbar macht.
- Bei 4 gleichzeitigen Anfragen ist das System beinahe vollständig ausgelastet, würde aber immer noch Kapazitäten für die FTP-Transfers besitzen.

Dieser Wert scheint sehr niedrig zu sein, er bedeutet aber für die Praxis bereits eine große Zahl von Zugriffen. Die Zahl bezeichnet exakt zeitgleich stattfindende Aufrufe der Webseiten oder Client Anfragen. Betrachtet man das Verhalten des Servers während der Stoßzeiten, so können pro Stunde also etwa 14.000 Seiten ausgeliefert werden. Da ein Benutzer nicht nur eine Seite, sondern mehrere in Folge aufrufen wird, können bei geschätzten 30-40 Seiten pro Sitzung etwa 3.000 unterschiedliche Anwender bedient werden. Dieser Wert sollte eventuell auf 2500 heruntergesetzt werden, um noch Ressourcen für die FTP-Transfers sowie weitere Aufgaben des Systems zur Verfügung zu haben.

Angesichts der Hardwareausstattung des Servers ist dies ein sehr guter Wert. Der Hauptgrund dafür liegt in der guten Indizierbarkeit der Daten. Während etwa bei einer Suchmaschine oder einer Forum-Software große Datenmenge durchsucht werden müssen, sind beim DIRA-System lediglich die Meta-Daten sowie einige Verwaltungs-Informationen in der Datenbank gespeichert.

Die Anzahl der möglichen Zugriffe bei Verwendung von aktueller Serverhardware lässt sich nur grob schätzen, da die Leistung des Systems von einer Vielzahl von Faktoren abhängt. Dazu zählen unter anderem:

- Zugriffsgeschwindigkeit auf die Festplatten

- Datenrate des verwendeten Bussystems
- Netzwerkanbindung
- Fähigkeit des Prozessors, große Datenmengen zu verarbeiten

Da ein Vergleich solcher Systeme sehr schwierig ist, gehen wir für die Abschätzung von einem stark vereinfachten Modell aus. Wir wollen im Folgenden die Leistung eines Pentium Celeron 2,4 GHz Prozessors mit der eines Pentium 4 Extreme Edition mit 3,46 GHz vergleichen. Bei letztem handelt es sich um einen hochgetakteten, bei Servern gerne eingesetzten, Intel Xeon Prozessor mit 2 MByte Level-2 Cache. Dieser stellt wohl den schnellsten zurzeit verfügbaren Prozessor für Endanwender dar. Die Messungen wurden von der renommierten Seite Tom's Hardware durchgeführt[134]. Dabei sind folgende Werte von besonderer Bedeutung:

	P4 Celeron 2,4 GHz	P4 EE 3,46 GHz
PC2004, Speichertransfer	2263 Punkte	6413 Punkte
PC2004, CPU	2900 Punkte	5397 Punkte
Sandra, Fließkommaoperation	18713 Punkte	37445 Punkte
WinRAR Archiv erstellen	4:04 Minuten	10:20 Minuten

Tabelle 4.13.2d – Leistungsdaten von P4 Celeron und P4 Extreme Edition

Dabei lässt sich erkennen, dass der Pentium 4 Extreme Edition Prozessor zwischen 1,8 bis 2,5 mal schneller als Celeron Prozessor rechnet. Unter der Voraussetzung, dass die restliche Hardware (Festplatten usw.) den Chip nicht ausbremst, so kann der beschriebene Highend-Server etwa 8000 Benutzer pro Stunde bedienen. Dabei ist bereits die höhere Taktfrequenz des gemessenen Celeron Prozessors (2,4 GHz statt 1,8 GHz) mit einbezogen.

[134] http://www.tomshardware.de/cpu/charts/index.html?model1=48&model2=12&chart=15

Die Preise für ein solches System sind allerdings so hoch, dass man stattdessen die Anschaffung von mehreren billigen Rechnern in Erwägung ziehen kann. Dies hätte weiterhin eine Erhöhung der Redundanz und somit eine Verbesserung der Verfügbarkeit zur Folge. Im Abs. 5.1 wird genauer auf ein solches System eingegangen.

4.13.3. Bandbreite des Netzwerkes

Bei einer steigenden Anzahl von Clients gilt es, die beschränkte Bandbreite der Netzwerkanbindung des Servers zu beachten. Die maximale Kapazität des Servers ist dabei von der Bandbreite der Client-Anbindung als auch der Intelligenz der Synchronisations-Software abhängig. Für die Simulation der Netzwerklast wurde zunächst innerhalb eines lokalen Netzwerks per FTP eine größere Datei (500 MB) vom Server übertragen und dabei die Übertragungsgeschwindigkeit gemessen. Die effektive Geschwindigkeit betrug dabei etwa 60 MBit/s, was für ein 100 MBit/s Netzwerk Vollauslastung bedeutet.

Diese Rate entspricht allerdings einem Idealwert, der bei Übertragungen über das Internet kaum zu erreichen ist. Wie erwähnt, verfügt der bei Level-3 untergebrachte Server über keine hochwertige Netzwerkverbindung. Die höchste in Tests erzielte Übertragungsrate betrug etwa 20 MBit/s (entsprechend 2,5 MByte/s). Bei einer Vielzahl von gleichzeitigen Transfers kann man wegen des zusätzlichen Overheads von einer Nettoübertragungskapazität von etwa 18 MBit/s ausgehen.

Das sich daraus ergebende Limit für die Anzahl der Clients ist davon abhängig, ob diese die Updates gleichzeitig oder versetzt durchführen. Sind diese etwa alle per DSL (1 MBit/s Anbindung) mit dem Internet verbunden, so kann der Server bei schlechtem Timing der Synchronisation nur circa 18 Clients gleichzeitig bei voller Bandbreite beliefern. Dieser Wert scheint niedrig zu sein, im Praxiseinsatz bewies er jedoch ausreichendes Leistungsvolumen, wie mit folgender Kalkulation deutlich gemacht wird:

Geht man von einem sehr aktiven Distributionssystem mit 30 neuen Musikstücken am Tag aus, dann benötigt ein Client bei der beschriebenen Anbindung etwa 15 Minuten, um diese zu übertragen. Legt man nun die Synchronisationsintervalle so, dass diese sich nicht überschneiden, so können pro Tag 1.728 Clients abgeglichen werden. Da eine so exakte Abstimmung in der Praxis wohl kaum zu erzielen ist, sollte man diesen Wert auf etwa 1.500 reduzieren. Es sei erwähnt, dass die Funktionalität des Systems bei zu hoher Netzwerklast nicht beeinträchtigt wird, lediglich der Zeitraum für die Übertragung vergrößert sich. Dieser Punkt stellt eine Verbesserung der Betriebsstabilität eines verteilten Radiosystems dar und unterstreicht somit These 9.

Betrachtet man die mögliche Anzahl von Clients beim Streaming von Audiodaten, so könnte der Server bei gegebener Anbindung bei 128 KBit/s Bitrate ca. 160 gleichzeitige Zuhörer beliefern, ohne dass es zu Aussetzern bei der Übertragung kommt.

Durch den Einsatz der Peer-to-peer-Übertragungstechnik ist es möglich, die Skalierbarkeit des Systems nochmals erheblich zu verbessern. Diese Verbesserungsmöglichkeit werden wir in Abs. 5.3 ausführlich behandeln.

5. Einsatz- und Verbesserungsmöglichkeiten

Das DIRA-System ist in der derzeitigen Form voll einsatzfähig, allerdings gibt es, wie bei Software dieses Umfangs üblich, noch reichlich Ansätze für Verbesserungen. Art und Umfang dieser Änderungen hängen stark vom geplanten Einsatzzweck ab, daher sollen in diesem Abschnitt beide Aspekte gemeinsam betrachtet werden. Will man das System professionell einsetzen, so sind die Punkte Betriebs- und Einbruchsicherheit von großer Bedeutung. So wäre etwa ein erfolgreicher Hacker-Angriff, bei dem alle Jingles durch veränderte Elemente ausgetauscht werden, eine Katastrophe für die Reputation des Betreibers.

5.1. Erhöhung der Verfügbarkeit

Egal in welcher Variante man das DIRA-System einsetzen will, eine Grundvoraussetzung für die professionelle Nutzung ist eine hohe Verfügbarkeit. Man mag meinen, dass ein Server mit redundanten Festplatten hinreichende Reserven bietet. Während der Entwicklung des Systems kam es allerdings zu insgesamt drei Ausfällen von unterschiedlicher Länge:

- Lüfterschaden: Dieses Problem trat innerhalb von etwa einem Jahr zwei Mal auf. Wegen der geringen Bauhöhe des Servers (etwa 2,5 cm) ist die Größe des Prozessor-Kühlkörpers stark limitiert – das hat zur Folge, dass sich der darauf montierte Lüfter sehr schnell drehen muss, um die Abwärme abzuleiten. Dies führt zu einer schnellen Abnutzung der Lager und letztendlich zum Ausfall des Lüfters. Glücklicherweise erkennt das Mainboard die viel zu hohe Prozessortemperatur und fährt den Rechner herunter, um Schäden an der CPU zu vermeiden.
- Festplattenschaden: Wie bereits erwähnt, sind in diesem System beide Festplatten innerhalb kürzester Zeit ausgefallen. Das ist natürlich nicht normal, es scheint sich dabei wohl um eine sehr schlechte Baureihe des Herstellers Maxtor zu handeln – dies wurde von einem Techniker des Housing-Providers bestätigt. Die Reparatur war hier wesentlich aufwendiger, glücklicherweise existierten Backups der Datenbank und der Webseiten, so dass lediglich die vorhandenen Titel verloren waren.

Die einfachere Herangehensweise an dieses Problem wäre der Einsatz von größeren Server-Gehäusen. Offensichtlich gibt es beim Abtransport der Wärme bei einer Höheneinheit ziemlich wenig Spielraum. Dabei muss man auch beachten, dass ein Gerät, welches auf dem heimischen Schreibtisch einwandfrei funktioniert, in ein Regal mit etwa 40 anderen Computern montiert wird. Diese produzieren ebenfalls viel Wärme, die sich auch auf das eigene System überträgt. Bei Gehäusen mit zwei

Höheneinheiten sind wohl mehr Reserven vorhanden.

Problematisch daran ist allerdings, dass die Miete für einen Server mit zwei Höheneinheiten (und einem Netzwerk-Port) nur etwas weniger kostet als zwei getrennte Höheneinheiten mit entsprechend zwei Ports. Verfügt man nun über die finanziellen Mittel, zwei 1HE Server zu kaufen, so kann man mittels vorhandener Software ein kostengünstiges Linux-Cluster bauen, dessen Verfügbarkeit als sehr hoch gelten darf. Man verfügt in diesem Fall über eine komplett redundante Hardware-Ausstattung, so dass der Ausfall jeder Komponente abgefangen werden kann. Im Netzwerk-Jargon spricht man dabei von der Eliminierung der „Single Point Of Failure(s)“[135].

Ein einfaches und im Vergleich zu einem Hardware-Cluster trotzdem zuverlässiges Failover-Cluster ist mit folgenden Komponenten zu realisieren[136]:

- Hardware: Wie bereits beschrieben sind zwei 1HE Server mit aktuellen Hardware-Komponenten für die meisten Anwendung ausreichend. Zusätzlich zum externen Netzwerk-Interface sollten sie über ein schnelles internes Interface (am besten Gigabit-Lan) zum Datenaustausch zwischen beiden Stationen verfügen.
- Software: Damit das Cluster im laufenden Betrieb und ohne Unterbrechung von einem Server zum anderen umschalten kann, ist es erforderlich, dass die Datenbestände auf beiden Geräten absolut identisch sind. Zu diesem Zweck wurde die Software *drdb*[137] entwickelt.
    - *drdb*: Dabei handelt es sich um ein Kernel-Modul, welches Schreibzugriffe auf eine festgelegte Festplatten-Partition über ein Netzwerk auf einen anderen Rechner dupliziert. Dabei stehen verschiedene Optionen bezüglich der Datenintegrität zur Verfügung. In der Regel wird man dabei auf Protokoll C zurückgreifen – in

[135] http://c2.com/cgi/wiki?SinglePointOfFailure

[136] http://linux-ha.org/

[137] http://www.drbd.org/

diesem Modus wird der Schreibvorgang erst abgeschlossen, wenn der zweite Rechner eine Bestätigung über den korrekten Abschluss des Vorgangs gesendet hat. Auf diese Weise lassen sich selbst Datenbanken mit sehr vielen Transaktionen duplizieren. Die Geschwindigkeit der Schreibzugriffe ist dabei stark von der Netzwerkgeschwindigkeit zwischen beiden Rechner abhängig.

- *Heartbeat*: Diese Software muss ebenfalls auf beiden Knoten des Clusters laufen. Sie überprüft kontinuierlich die Funktionalität der Rechner. Momentan wird dabei nur eine einfache Aktiv/Passiv Implementierung unterstützt, das heißt, ein Rechner des Clusters nimmt aktiv die Netzwerkanfragen entgegen, während der andere im Standby-Betrieb ohne weitere Funktion läuft. Stellt *Heartbeat* fest, dass der aktive Server nicht mehr erreichbar ist, so wird auf dem passiven Server eine Reihe von (frei definierbaren) Befehlen ausgeführt, so dass dieser die Funktionen des ausgefallenen Rechners übernimmt. *Heartbeat* überprüft dabei nur, ob ein Rechner erreichbar ist, nicht aber, ob ein bestimmter Dienst (z.B. der Webserver) noch funktioniert.
- *Mon*: Zu diesem Zweck benutzt man Mon. Mit dieser Software kann man bestimmte Dienste überwachen. Wird festgestellt, dass etwa der Webserver nicht mehr verfügbar ist, so signalisiert *Mon Heartbeat*, dass es auf den anderen Server umschalten soll.
- *Fake*[138]: Damit der passive Rechner die Rolle des Aktiven übernehmen kann, macht man Gebrauch von virtuellen IP-Adressen. Neben der primären Adresse des Servers benötigt man eine zweite (ebenfalls öffentlich zugängliche) virtuelle IP-Adresse. Diese wird von sämtlichen Diensten zur Kommunikation mit dem Netz benutzt. Fällt der aktive Node aus, so startet *Heartbeat* auf dem bisher passiven Gerät das Programm *Fake*, welches die virtuelle Adresse des ausgefallenen Gerätes übernimmt.

[138] http://www.vergenet.net/linux/fake/

Ein Cluster mit diesen Software-Elementen würde sich schematisch folgendermaßen darstellen:

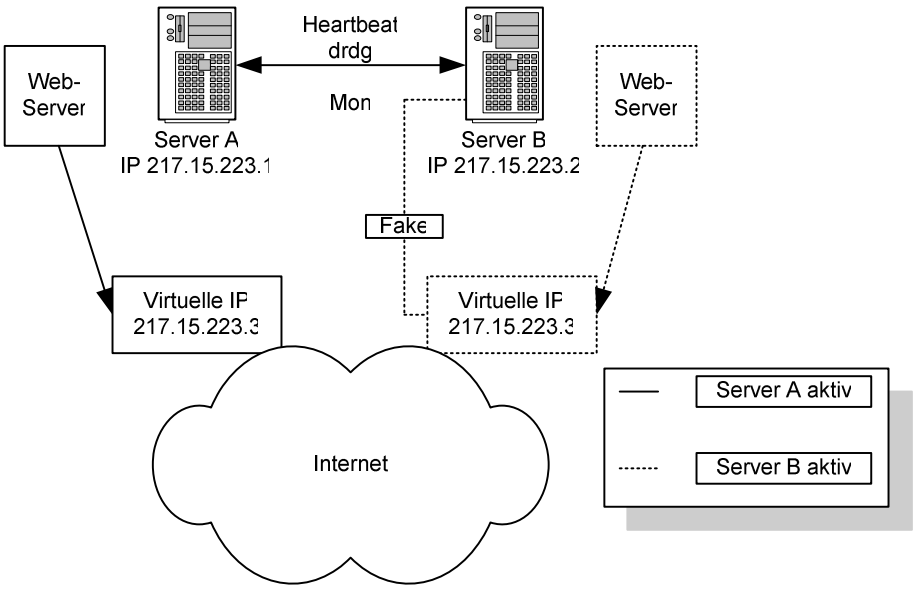

Abbildung 5.1a – Schema eines Linux Failover-Clusters

Im Normalbetrieb nimmt die auf Server A laufende Webserver-Software Anfragen auf dem virtuellen Interface 217.15.224.3 an. Stellt das auf Server B laufende Heartbeat-Programm fest, dass Server A nicht mehr verfügbar ist, so benutzt es *fake*, um die virtuelle Adresse zu übernehmen und startet daraufhin den Webserver. Der Zeitrahmen für die Übernahme sollte nicht mehr als 10 Sekunden betragen, so dass der Server nur für sehr kurze Zeit nicht verfügbar ist[139].

Es sei noch erwähnt, dass diese Art von Cluster keinerlei Lastverteilungs-Algorithmen mit sich bringt. Will man das System für eine sehr große Zahl von Nutzern (z.B. mehrere tausend Anwender greifen gleichzeitig auf die Webseite und die Datenbank zu) hochverfügbar machen, so muss man dafür auf Projekte wie etwa das „Linux Virtual Server Project“[140] zurückgreifen. Die Anzahl der benötigten Rechner erhöht sich dadurch auf mindestens vier.

---

[139] [SOM2002], S. 39 - 42

[140] http://www.linuxvirtualserver.org/

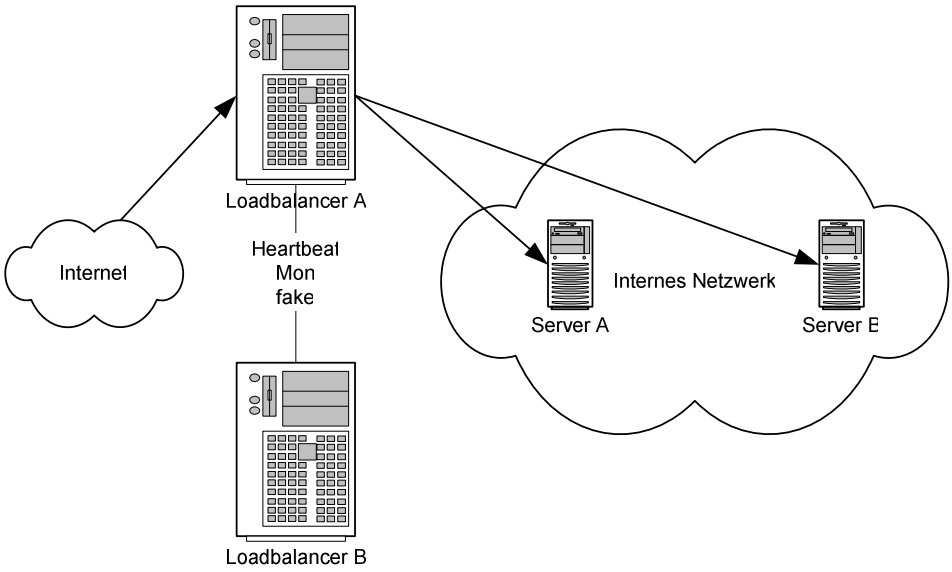

Abbildung 5.1b – Cluster mit Loadbalancing-Funktion

Dabei arbeiten die Rechner Loadbalancer A und B wie bereits beschrieben als Failover-Cluster. Sie überwachen mit Hilfe des *Mon* Programms sowohl die Funktion des anderen Cluster-Mitgliedes als auch die von Server A und B. Weiterhin kommt die Software *Director*[141] zum Einsatz, welche Anfragen an die Loadbalancer an Server A oder B weiterleitet. Dabei lässt sich einstellen, dass etwa Rechner A, der über eine bessere Hardwareausstattung verfügt, mehr Anfragen bekommt als Rechner B. Natürlich ist auch die Installation von mehr als zwei Servern möglich, große Firmen benutzen dabei oft mehrere hundert Maschinen[142].

Für die später in diesem Kaptitel beschriebenen Anwendungen sollte aber die einfache Failover-Lösung ausreichen.

[141] http://www.linuxvirtualserver.org/HighAvailability.html

[142] [SOM2002], S. 148 - 155

5.2. Erhöhung der Systemsicherheit

Unter diesem Punkt sollen verschiedene Maßnahmen zur Sicherung des Systems vor unbefugtem Zugriff beschrieben werden. Da ein Server voller Musikstücke einen nicht unerheblichen Reiz auf böswillige Internet-Benutzer ausübt, sollte man hier alle verfügbaren Mittel einsetzen, auch wenn manche davon übertrieben erscheinen. Bei der Erhöhung der Systemsicherheit gilt als oberstes Ziel, die möglichen Angriffsflächen zu verkleinern.

5.2.1. Webserver

Das Computer Emergency Response Team der Universität Carnegie Mellon[143] analysiert seit geraumer Zeit Angriffe auf Computersysteme. Dabei wurde festgestellt, dass eine Vielzahl von erfolgreichen Einbrüchen über den Webserver erfolgt. Der Grund dafür ist nahe liegend: Durch den Siegeszug von Skriptsprachen für dynamische Webseiten (z.B. *PHP*, *ASP*) und entsprechenden Frameworks (etwa *Typo3*[144] oder *Coocon*[145]) steht einem Angreifer eine vollständige Programmiersprache, welche noch dazu über eine Webseite abgefragt werden kann, zu Verfügung. Das bedeutet natürlich nicht, dass es jedem Benutzer möglich wäre, eigene Programme auf einem fremden Webserver auszuführen. Vielmehr kann man unter Umständen Fehler in den vorhandenen Skripten ausnutzen, um eigenen Code einzuschleusen. Dazu zwei Beispiele:

- url-fopen Fehler: Der Befehl fopen() wird in *PHP* dazu benutzt, eine Datei zu öffnen, um daraus Daten (etwa Benutzerinformationen) zu lesen. So wird zum Beispiel der Datei load.php?config.txt über den Wert nach dem Fragezeichen der zu ladende Dateinamen übermittelt. Passt der Programmierer nicht auf, so ist es möglich, durch den Aufruf von etwa

[143] http://www.cert.org

[144] http://www.typo3.net

[145] http://cocoon.apache.org

load.php?http://www.angreifer.de/Skript.php die Datei Skript.php von einem entfernten Server zu laden und unter dem lokalen Sicherheitskontext auszuführen.

- Shell Escape: Bei dieser Art von Angriffen wird versucht, Befehle in die Parameter eines Programmaufrufs einzuschleusen. Das kann gelingen, wenn das bearbeitende Skript Sonderzeichen in etwa einem Dateinamen nicht richtig „escaped", also unschädlich macht. Würde man zum Beispiel als Dateinamen „egal;ls –lr" angeben, so kann es passieren, dass ein externes Programm zuerst die vorgesehene Operation mit der Datei egal macht, danach aber durch den „ls –lr" Befehl den Inhalt des aktuellen Verzeichnisses ausgibt, da ein Semikolon unter Linux dazu dient, Befehle zu trennen[146].

Es gibt natürlich noch eine Vielzahl von weiteren Angriffsarten, auf die aber hier nicht eingegangen werden soll.

Wie in Abs. 5.2.1 beschrieben arbeitet der *Apache* Server (zumindest in der Version 1) mit einem fork-Prozessmodell. Dabei erzeugt ein Master-Prozeß (mit vollen Administratorrechten) eine Reihe von Unterprozessen, welche wiederum die Web-Anfragen entgegennehmen. Diese Unterprozesse laufen unter einem speziellen Benutzerkonto (meist wwwrun und *Apache*) mit eingeschränkten Rechten. Dies äußert sich zum Beispiel daran, dass dieser Benutzer keine Einstellungen am System ändern kann. Gelingt es einem Angreifer also, etwa durch einen Fehler in einem *PHP*-Skript, eigenen Code einzuschleusen, so kann er damit zwar noch nicht den Server komplett übernehmen, hat aber immerhin schon Zugriff auf alle für den Benutzer wwwrun zugänglichen Dateien.

War der Administrator des Servers nun nachlässig und hat wichtige Sicherheitsupdates nicht eingespielt, so könnte der Angreifer durch Ausführen eines so genannten Root-Exploits[147] den Rechner komplett übernehmen. Ein Beispiel für

[146] [ANO1999], S. 401 - 407

[147] http://www.rootforum.de/forum/viewtopic.php?t=19898

einen solchen Exploit war etwa der Fehler in der do_brk() Routine des Linux-Kernels, der zur Kompromittierung des Debian-Webservers führte[148].

Das klingt relativ kompliziert, es existieren auf einschlägigen Seiten aber bereits vorgefertigte Skripte, welche einen Fehler in einer bestimmten Software automatisch ausnutzen, so dass der Hacker nur noch ein Programm mit einer Zieladresse starten muss. Die Anwender dieser Software werden häufig abwertend „Skript Kiddies[149]“ genannt – in Unterscheidung zu „richtigen“ Hackern, welche versuchen ein System durch eigene Bemühungen zu übernehmen.

Hat ein Angreifer ein eingeschränktes Benutzerkonto übernommen, so könnte er im Falle des DIRA-Systems bereits auf die gespeicherten MP3-Dateien zugreifen. Dies ist dadurch bedingt, dass der Webserver die hochgeladenen Musiktitel in einem Verzeichnis speichern muss, für welches er natürlich Zugriffsrechte benötigt. Weiterhin gilt, dass der Webserver auch auf Teile des Systems zugreifen können muss, um etwa Protokoll- oder Konfigurationsdateien zu lesen. Für das Problem, dass der wwwrun-Benutzer auf viele Teile des Systems zugreifen kann, existiert eine Reihe von Lösungen.

Die einfachste Methode, Zugriffe auf fremde Verzeichnisse zu unterbinden, sind die Einstellungen „safe_mode on“ sowie „open_basedir /var/www/“ in der Datei php.ini[150]. Sie bewirken, dass *PHP* (unter anderem) keine Systembefehle mehr ausführt und außerdem nur noch Dateien unterhalb des Verzeichnisses /var/www öffnet.

Leider eignet sich diese Methode nicht für das DIRA-System, da vor allem beim Upload von Dateien häufig Gebrauch von Systembefehlen gemacht wird – etwa zum Kodieren der MP3-Dateien oder zur Fehlerüberprüfung. Zwar wäre die Nutzung von open_basedir auch ohne den Safemode möglich, allerdings hat man auch hier das Problem, dass die Audio-Dateien innerhalb des Webserver-

[148] http://www.debian.org/security/2003/dsa-403

[149] http://www.webopedia.com/TERM/S/Skript_kiddie.html

[150] http://de2.php.net/features.safe-mode

Verzeichnises gespeichert werden müssen. Sollte sich in den *PHP*-Skripten ein Fehler befinden, so wären die Dateien eventuell über einen Webbrowser zum Download verfügbar. Weiterhin gilt die open_basedir Einschränkung nur für Dateiaufrufe durch *PHP*, Befehle an Systemprogramme sind dadurch nicht gebunden.

Um diese Sicherheitslücke zu schließen, kann das Programm *suPHP*[151] benutzt werden. Es ermöglicht die Ausführung von *PHP* unter einem anderen Benutzerkonto als wwwrun. Dabei wird *PHP* nicht mehr als Modul direkt in *Apache* geladen, sondern als ausführbare Datei (Paket php-cgi in Debian) über den *suPHP*-Wrapper gestartet. Dieses Programm bewirkt, dass *PHP* unter dem Benutzer- und Gruppenkonto des aufgerufenen Skripts gestartet wird. Durch das Setzen von sehr restriktiven Berechtigungen für diesen Benutzer kann man weite Teile des Servers ausblenden.

Damit ist allerdings noch nicht das Problem der MP3-Dateien im Webordner gelöst. Eine etwas aufwendigere Lösung hierzu wäre, neue Uploads nicht direkt in das Archiv zu schreiben, sondern erst in einem temporären Verzeichnis (unterhalb des Web-Basisverzeichnisses) zu speichern. Ein im Hintergrund laufender *Cron* Auftrag (mit mehr Rechten als der Webserver) überprüft jede Minute, ob neue Dateien vorhanden sind und verschiebt diese bei Bedarf in das eigentlich vorgesehene Archiv-Verzeichnis. Analog zu diesem Verfahren ist das Löschen von Titeln zu realisieren.

[151] http://www.suphp.org/Home.html

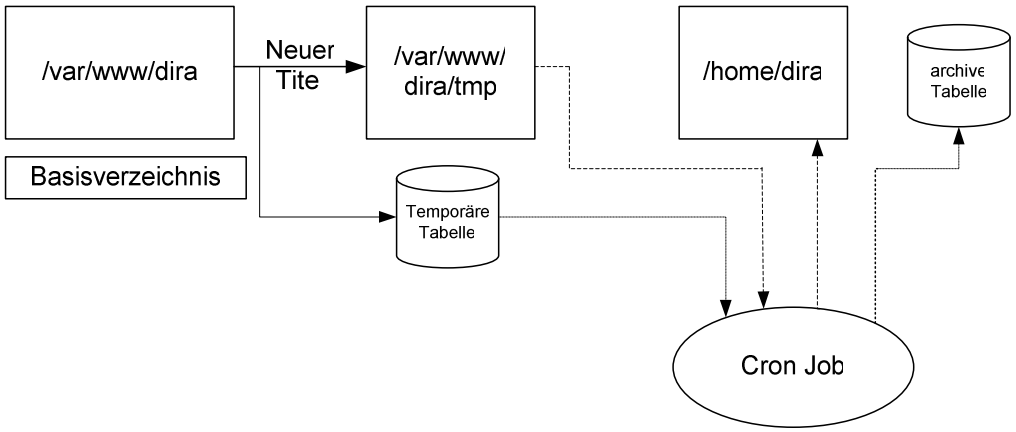

Abbildung 5.2.1a – Entfernen der Musiktitel aus Webverzeichnis per Cron-Job

Dieses Archiv-Verzeichnis ist so geschützt, dass der Webserver über keinerlei Zugriffsrechte verfügt.

So abgesichert sollten die meisten Angriffe der „Skript Kiddies“ ins Leere laufen. Zwar ist dieser Ansatz auch nicht perfekt, da der Webserver noch immer nicht in einer chroot Umgebung läuft (dann wäre er komplett vom restlichen System gekapselt), die Tatsache, dass viele Internet-Hoster (z.B. 1&1 oder Strato) ebenfalls die *suPHP* Konfiguration einsetzen, um ihre Server abzusichern, spricht allerdings für deren Robustheit.

Eine chroot-Umgebung für den *Apache* Webserver ist mit dem mod_security[152] - Paket zu realisieren. Leider ist das Programm zurzeit offensichtlich noch nicht stabil genug, da bei Tests häufig „Error 500“ Fehler auftraten. Weiterhin stürzte die *Apache*-Software mehrmals bei Einsatz des Moduls ab. Die Programmierer raten auf der Webseite von der Verwendung in Produktionssystemen ab. Generell gilt es aber, diese Software im Auge zu behalten, da hiermit ein wirklich sehr sicherer Webserver realisiert werden kann.

[152] http://www.modsecurity.org/

5.2.2. Verschlüsselung der Webseiten

Ein Nachteil bei den im DIRA-System stattfindenden FTP- und *MySQL*-Verbindungen ist, dass dabei sämtliche Daten unverschlüsselt über das Netz gehen. Zwar ist das Belauschen von Internet-Verbindungen alles andere als trivial (der Angreifer muss Zugriff auf die Netzwerk-Infrastruktur haben oder sich im gleichen Subnetz befinden), komplett ausschließen kann man es aber nicht. Auch muss man in Betracht ziehen, dass die Festplatte eines Clients gestohlen und nach vorhandenen Passwörtern durchsucht wird. Aus diesem Grund ist es sinnvoll, sämtliche Kommunikation sowohl zwischen Client und Server als auch Server und Anwender zu verschlüsseln.

Diese Art der Sicherung lässt sich sehr einfach realisieren. Alle *PHP* Skripten sind Secure Socket Layer (SSL) tauglich. Um diese Art der Verschlüsselung zu nutzen genügt es, das Debian Paket libapache-mod-ssl zu installieren und das mod_ssl Paket in der *Apache*-Konfigurationsdatei zu aktivieren. Daraufhin lässt sich das System mittels des https:// Präfixes verschlüsselt nutzen. Um die Konfiguration zu komplettieren, muss allerdings ein Zertifikat bei einer Registrierungsstelle wie etwa VeriSign[153] oder GeoTrust[154] erworben werden, mit welchem der Server sich einwandfrei „ausweisen“ kann. Da dies mit Kosten (ca. 150.- € pro Jahr) verbunden ist, wurde auf diese Art der Verschlüsselung verzichtet.

5.2.3. Verschlüsselung der MySQL-Verbindung

Die Verschlüsselung des *MySQL*-Datenstroms ist theoretisch kein Problem, allerdings traten bei der praktischen Umsetzung Probleme auf, die sich nicht einwandfrei lösen ließen. Das Kommunikationsmodell zwischen Server und Client sieht im Normalfall (also ohne Verschlüsselung) folgendermaßen aus:

---

[153] http://www.verisign.com/products/site/index.html

[154] http://www.geotrusteurope.com/de/

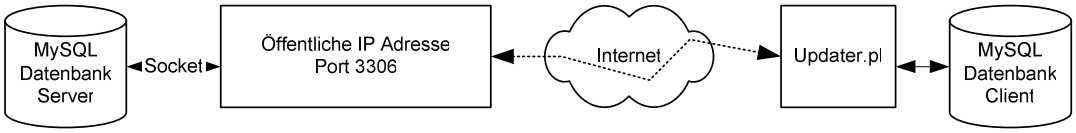

Abbildung 5.2.3a – Unverschlüsselte MySQL Kommunikation

Dabei wartet die *MySQL*-Datenbank des Servers auf einem festgelegten Port (in der Regel 3306) auf eingehende Verbindungen. Will das *Updater.pl*-Modul zum Beispiel feststellen, ob neue Titel vorhanden sind, dann baut es eine Verbindung zu diesem Port auf, authentifiziert sich mit Benutzernamen und Passwort und kann daraufhin auf die Datensätze des Servers zugreifen. Will man nun diese Kommunikation verschlüsseln, so kann man dafür auf das Programm *stunnel*[155] zurückgreifen. Dieses erlaubt es, einen „Secure Tunnel" zwischen zwei Rechnern aufzubauen. Folgende Einstellungen sind dabei notwendig:

- Serverseitig: Der *MySQL*-Datenbankserver wird dahingehend konfiguriert, dass er keine Anfragen auf der öffentlichen IP-Adresse mehr entgegen nimmt, sondern nur noch auf dem internen Loopback-Device (Interface lo). Mit dem Programm *stunnel* wird anschließend ein Tunnel zwischen dem *MySQL*-Port des Loopback-Devices und einem beliebigen Port (z.B. 3307) des öffentlichen Interfaces aufgebaut.
- Clientseitig: Auch hier wird ein Tunnel ausgeführt. Der lokale Endpunkt ist wieder ein beliebiger Port (z.B. 5001) auf dem Loopback-Device, der entfernte Endpunkt ist in unserem Beispiel Port 3307 des Servers.

[155] http://www.stunnel.org

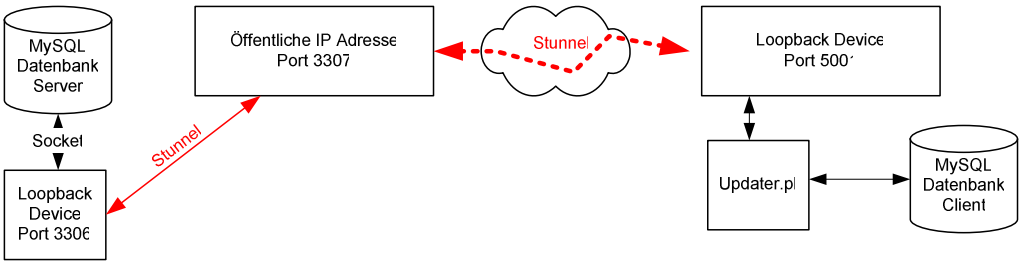

Abbildung 5.2.3b – Verschlüsselte MySQL Verbindung

In der Grafik wird deutlich, dass sowohl Client als auch Server glauben, nur auf dem lokalen Interface mit der Datenbank zu kommunizieren. Sämtliche Transfers über den Tunnel laufen komplett transparent ab.

Zusätzlich zur Verschlüsselung des Datenstroms kann *stunnel* auch noch eine Authentifizierung über Zertifikate ermöglichen. Dabei stehen verschiedene Optionen zur Verfügung:

- Serverseitige Authentifizierung: Dabei wird auf dem Server ein privater und ein öffentlicher Schlüssel erstellt. Der öffentliche Schlüssel wird an die Clients verteilt. Beim Verbinden mit dem Server vergleichen diese den gespeicherten mit dem vom Server angebotenen Schlüssel. Stimmen beide überein, so ist sichergestellt, dass es sich um den richtigen Server handelt.
- Server- und clientseitige Authentifizierung: Zusätzlich zu den gerade beschriebenen Überprüfungen werden hierbei auf dem Server für jeden Client Zertifikate gespeichert. Damit ein Client Zugriff erhält, muss er neben dem öffentlichen Schlüssel auch das vom Server ausgestellte Zertifikat besitzen.

Da die Verwaltung der Client-Zertifikate auf dem Server nur recht aufwendig zu realisieren ist (die Verwaltung soll über den Webserver geschehen, man will aber diesem definitiv keinen Zugriff auf das Verzeichnis mit den Zertifikaten geben), wurde in den Tests nur die erste Variante benutzt.

Dabei konnte erfolgreich ein Tunnel zwischen Client und Server hergestellt werden, auch der Austausch von Datensätzen funktionierte einwandfrei. Probleme bereiteten allerdings die Wartezeiten während des Dateidownloads. Da während dieser Zeit keine *MySQL*-Daten ausgetauscht werden, scheint *stunnel* die Verbindung für tot zu erklären und beendet sie. Will der Client nach den Downloads wieder auf den Server zugreifen, so scheitert er am nicht mehr vorhandenen Tunnel. Vermutlich lässt sich dieses Problem entweder durch Optimierung der Datenbankanfragen oder durch Neukompilieren von *stunnel* (mit längeren Timeout-Zeiten) lösen.

5.2.4. Verschlüsselung der FTP-Verbindung

Die Verschlüsselung der FTP-Verbindung stellt auf Grund der Struktur des FTP-Protokolls das größte Problem dar. Dabei werden im Unterschied zu *MySQL* zwei verschiedene Kommunikationskanäle benutzt. Zum einen benutzt der FTP-Client einen Kontroll-Kanal zur Übermittlung der Befehle (z.B. Download starten), zum anderen einen Daten-Kanal zur eigentlichen Übertragung der Dateien. Serverseitig sind dabei die Ports (21 und 20) festgelegt, der Client wählt für sich einen beliebigen Absenderport (Portnummer > 1024) aus[156].

Es ist vermutlich möglich, beide Kanäle mit Hilfe von *stunnel* zu verschlüsseln, eine einfachere Lösung bietet aber FTP-TLS. Dabei handelt es sich um eine Mischung aus FTP- und SSL Verschlüsselung[157]. Es werden nur die Daten des Kontroll-Kanals (also auch der Benutzername und das Passwort) verschlüsselt, die Nutzdaten bleiben unverschlüsselt. Für das DIRA-System sollte dies aber ausreichend sein, da das Extrahieren von MP3-Dateien aus dem Datenstrom sehr aufwendig ist.

[156] http://slacksite.com/other/ftp.html

[157] http://www.ietf.org/internet-drafts/draft-murray-auth-ftp-ssl-13.txt

Realisieren lässt sich dieses Verfahren, indem man den benutzten *PureFTP*-Server mit der Option „—with-tls“ kompiliert[158]. Startet man das Programm daraufhin mit der Option „—tls=2“, so akzeptiert der Server nur noch verschlüsselte FTP-Verbindungen.

Das Problem bei dieser Lösung ist, dass es in *Perl* noch kein Modul für FTP-TLS Verbindungen gibt. Zwar ließe sich dies wohl mit einer Hilfskonstruktion aus externen Programmen lösen (z.B. *lftp*[159]), man verliert dann aber im *Updater.pl*-Skript die direkte Kontrolle über den Transfer. Da dieses Art der Verschlüsselung noch Entwurfstatus hat, wird es wohl noch einige Zeit dauern, bis sie in *Perl* implementiert ist.

5.3. Verbesserung der Erfolgsrate durch Peer-to-peer Transfers

Wie in Abs. 4.6 gezeigt, bietet die verwendete Hardware genügend Reserven für eine hohe Anzahl von Benutzern. Der limitierende Faktor ist ab einer gewissen Anzahl von Clients die Netzwerkanbindung des Servers. Neben der offensichtlichen Lösung, diesen etwa mit einem Gigabit-Netzwerkadapter auszustatten, besteht durch den Einsatz von Peer-to-peer-Übertragungen zwischen den Clients die Möglichkeit, den Traffik besser zu verteilen. Die einzige zusätzliche Anforderung ist dabei, dass die Clients nicht per ISDN sondern über eine permanente Netzwerkverbindung verfügen.

Die folgenden Überlegungen sind theoretischer Natur, da leider weder die Hardware noch eine entsprechend große Anzahl von Testpersonen für die Realisierung von Peer-to-Peer-Downloads zur Verfügung standen. Es ist zum Glück nicht notwendig, sämtliche Prozeduren für diese Art von Downloads selber zu implementieren, man kann dabei auf eine Reihe von bereits vorhandenen Programmen zurückgreifen.

---

[158] http://www.pureftpd.org/README.TLS

[159] http://lftp.yar.ru/

Am vielversprechendsten scheint dabei die Software *BitTorrent*[160] zu sein. Sie ist frei verfügbar und sowohl Client als auch Server stehen unter Linux zur Verfügung. Im Unterschied zu Programmen wie *eMule* oder *Kazaa* verwendet sie keine eigene Benutzeroberfläche für die Suche nach Dateien sondern greift auf einen vorhandenen Webserver und Browser für diese Aktion zurück. Für das DIRA-System würde der Umstieg auf dieses Protokoll folgende Veränderungen mit sich bringen:

Serverseitig:

- Auf dem Server wird eine so genannte Tracker-Software installiert. Sie stellt den Mittelpunkt des Peer-to-Peer-Netzes dar. Deren Aufgabe besteht darin, Verbindung auf die in den Meta-Dateien festgelegten Freigaben zu überwachen. Anders als bei etwa *eMule* stehen verschiedene Tracker nicht miteinander in Verbindung.
- Zusätzlich wird auf dem Server die Client-Version der Software installiert, über welche die vorhandenen MP3-Dateien freigegeben werden.
- Um Dateien für den Download freizugeben, wird eine Meta-Datei (Endung .torrent) benötigt. Diese enthält neben dem vorgeschlagenen Dateinamen, der Adresse des Trackers auch noch MD5 Prüfsummen zur Integritätsüberprüfung.
- Die Meta-Dateien werden entweder in einem passwortgeschützen Bereich der Webseite bereitgestellt oder sie werden, da sie in der Regel nicht größer als ein Kilobyte sind, über die *MySQL*-Verbindung an die Clients verteilt. Dabei können auch mehrere zu übertragende Dateien in einer Metadatei zusammengefasst werden.

Clientseitig:

- Hier wird, wie auf dem Server, die Client-Version der *BitTorrent*-Software installiert.

[160] http://bitconjurer.org/BitTorrent/

Wird durch den Anwender nun etwa ein neuer Musiktitel auf den Server eingespielt, so erfährt der Client davon wie bisher über die Timestamps der *MySQL*-Datenbank. Im Unterschied zum bisherigen System lädt er sich die Datei dieses Titels aber nicht per FTP-Protokoll auf die lokale Festplatte, sondern übergibt der *BitTorrent*-Client-Software die entsprechende Meta-Datei. Der darauf folgende Mechanismus erklärt den Namen der Software (Torrent = reißender Strom, Sturzbach).

Der erste Client der die Datei anfragt, kann diese natürlich nur vom Server übertragen, da sie ja auf sonst noch keinem Rechner vorhanden ist. Startet daraufhin Client Nummer Zwei seine Anfrage, so erkennt der auf dem Server laufende Tracker, dass die gewünschte Datei sowohl auf dem Server als auch auf Client Eins vorhanden ist. Daher wird Client Zwei die gewünschten Daten von beiden Rechnern übertragen. Dies funktioniert sogar, wenn der erste Client die Datei noch nicht vollständig empfangen hat (dann wird nur der bereits vorhandene Teil gesendet), auch mit abgebrochenen Downloads kann diese Software umgehen.

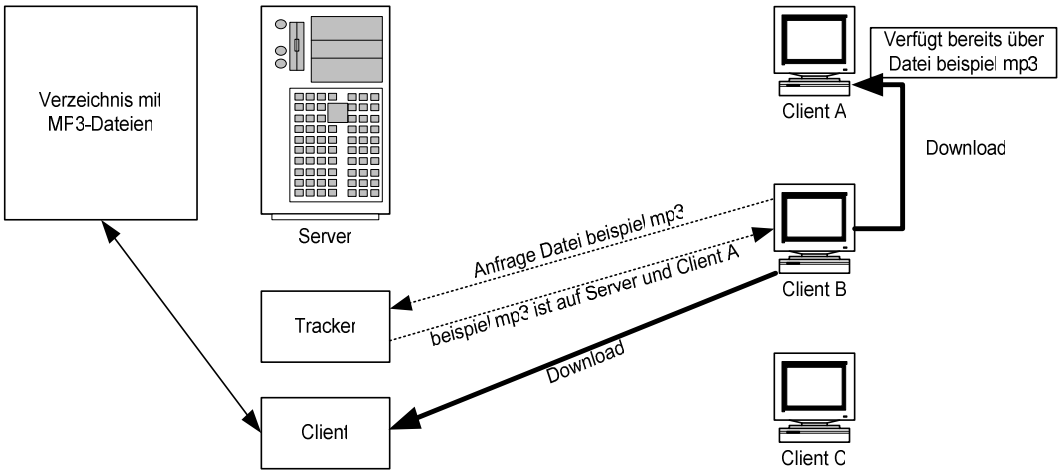

Abbildung 5.3a – Übertragung eines Titels durch den 2. Client

Man kann sich vorstellen, dass ab einer gewissen Sättigung der Clients fast keine Nutzdaten mehr vom Server gesendet werden müssen, die Clients machen die Transfers untereinander aus. Dabei ist einstellbar, wie lange sie einen erhaltenen

Titel freigeben. In der Regel dürfte ein Zeitraum von 24 Stunden ausreichend sein, da innerhalb dieser Zeit wohl alle vorhandenen Clients ein Update durchführen. Aber auch das Hinzufügen eines neuen Clients, der nur über einen Teilbestand der Daten verfügt, stellt kein Problem dar – er kann die benötigten Titel immer noch vom Server übertragen.

Die Verringerung des Übertragungsvolumens des Servers ist stark von der Anbindung der Clients und vom Timing der Synchronisation abhängig. Im schlechtesten Fall führen alle Clients gleichzeitig ein Update durch, dies würde bedeuten, dass der Server als einzige Quelle zur Verfügung steht und somit keine Verringerung des Traffiks erzielt werden kann. Im Idealfall wiederum beginnt zum Beispiel Client 2 erst mit dem Update, nachdem Client 1 mindestens 50 % einer Datei übertragen hat. In diesem Fall müsste der Server für Client 2 die Hälfte der Daten übertragen, für Client 3 nur 33 % usw. Für diesen Fall ergibt sich folgende Formel:

$$Transfervolumen = \sum_{Client} \frac{Datenmenge}{Client}$$

Formel 5.3a – Idealfall bei der Peer-to-peer Übertragung (Datenmenge am Server)

Neben der Einschränkung durch das Timing der Updates dürfte auch die Tatsache, dass die Clients über eine eingeschränkte Bandbreite verfügen (und somit nicht beliebig viele weitere Clients versorgen können), dazu führen, dass dieser Idealfall nicht erreicht werden kann. Im Vergleich zu Formel 4.2.2b sollte jedoch auf jeden Fall eine Reduzierung der zu übertragenen Datenmenge feststellbar sein, was eine Bestätigung für These 7 darstellt.

5.4. Einsatz als Beschallungssystem

Auf Grund der geringen Anforderungen an die Client-Hardware und der Unabhängigkeit vom Netzwerk würde sich das System stark für den Einsatz im

Bereich der Beschallungssysteme anbieten. Folgende Zielgruppen sind denkbar:

- Filialketten: Firmen, welche über ein ausgedehntes Netzwerk von Filialen verfügen, können das System nutzen, um eine einheitliche Beschallung zu erzielen. Dabei stehen ihnen auch Möglichkeiten zur zentralen Planung von Werbung zur Verfügung, um etwa Sonderangebote akustisch zu unterstützen.
- Restaurants und Bars: Bei dieser Zielgruppe steht wohl weniger die Planung eines eigenen Radiosenders im Vordergrund, sondern vielmehr die Möglichkeit, über eine unkomplizierte und automatisch aktualisierte Form der Beschallung zu verfügen.
- Große Unternehmen oder Institute: Wollte etwa die TU Berlin ein universitätsweites Campus-Radio realisieren, so würde das Streaming der Sendung eine erhebliche Belastung für das Netzwerk darstellen. Mittels des DIRA-Systems wäre eine Distribution des Audio-Materials (auch an geographisch verteilte Institute) leicht möglich.

Im Folgenden wollen wir die Veränderungen am Systems, die für einen solchen Einsatz nötig wären, betrachten.

5.4.1. Differenzierte Benutzerrechte

In der derzeitigen Form kennt das System nur eine Ebene von Benutzerrechten. Im Hinblick auf den Aufbau von professionellen Radiosystemen bietet sich eine Unterteilung in drei verschiedene Benutzergruppen an:

- Administrator: Diese Gruppe von Benutzern soll den einwandfreien Betrieb des Systems sicherstellen. Dazu müssen sie nicht nur kompletten Zugriff auf alle vorhandenen Daten erhalten, auch die Entwicklung einer Fehler-Tracking Webseite (z.B. Client 4 von Anwender user1 hat seit 24 Stunden kein Update durchgeführt) ist sinnvoll. Weitere Aufgaben der

Administratoren sind das Verwalten der Benutzer sowie der angeschlossenen Clients.

- Redakteure: Die Benutzergruppe erhält die Rechte, Musik und Programmelemente für alle Benutzer zu verwalten und gegebenenfalls neue Titel einzuspielen. Sie können allerdings keine neuen Anwender oder Clients anlegen.
- Anwender: Sie können, wie die Musikredakteure, Lieder und Programmelemente einspielen, allerdings nur für ihr eigenes Programm. Sinnvoll wäre auch eine detaillierte statistische Auswertung der gespielten Elemente, um so eine bessere Vermarktung von Werbezeiten möglich zu machen.

Bei dieser Art der Privilegientrennung werden folgende Annahmen vorausgesetzt:

- Der Betreiber des Systems stellt die Plattform zur Distribution von Audio-Inhalten für mehrere Anwender zur Verfügung. Das bedeutet, dass die vorhandenen Musikstücke für alle Nutzer auf dem zentralen Server gespeichert sind. Dies erleichtert zum einen die Wartung, zum anderen spart es natürlich auch viel Zeit, einen neuen Titel nur ein Mal einzuspielen.
- Die Anwender sind durch die Authentifizierung voneinander getrennt. Dabei steht jedem eine Webseite zur Verfügung, auf der er nach Anmeldung die Gestaltung seines Programms vornehmen kann. Dabei ist darauf zu achten, dass die Elemente verschiedener Anwender in unterschiedlichen Verzeichnissen gespeichert werden.

Durch diese Unterteilung hat man den Vorteil, dass dem Benutzer ein individuell konfigurierbares Radio-Automationstool zur Verfügung steht, während der Betreiber auf der anderen Seite alle vorhandenen Benutzer leicht mit neuer Musik versorgen kann.

5.4.2. Individuelle Datenbanken

Für einen kommerziellen Betreiber eines solchen Dienstes wäre es sehr abträglich, wenn plötzlich Werbespots von Nutzer A bei Nutzer B laufen würden. Daher ist es sinnvoll, den Datenbestand einzelner Kunden zu separieren, um so bereits auf Ebene der Datenbank den Zugriff auf falsche Daten zu unterbinden. Dabei sollen alle benutzer-spezifischen Informationen in die Benutzerdatenbank geschrieben werden, während globale Daten weiterhin in einer zentralen Datenbank gespeichert sind.

In der globalen Datenbank sind folgende Tabellen vorhanden:

- archive: Da das Musikarchiv für alle Benutzer zur Verfügung stehen soll, wird diese Tabelle öffentlich zugänglich angelegt.
- updated bzw. deleted: Damit die Clients über Veränderungen des Archivs informiert werden, müssen diese Tabellen ebenfalls global zugänglich sein.
- admins, musikred, users: Diese Tabellen enthalten die vorhandenen Administratoren, Musikredakteure und Benutzer.
- adminlog, musikredlog, userlog: Hier sind die Cookies der angemeldeten Benutzer gespeichert.
- news: Da der Nachrichtenticker systemweit zu lesen sein soll, ist er ebenfalls im öffentlichen Bereich gespeichert.

Neben der globalen Datenbank existiert für jeden Anwender des Systems eine eigene Datenbank, in welcher folgende Tabellen existieren:

- ad_archive, feature_archive, jingle_archive: In diesen Tabellen werden die Programmelemente des jeweiligen Benutzers gespeichert. Ihr Aufbau ist identisch zu den in Abs. 4.7.1 beschriebenen Strukturen.
- planned_songs, planned_songs_list: Hier werden die individuellen Playlists des Anwenders gespeichert.

- updated, deleted: Diese Tabellen enthalten die Ids von aktualisierten und gelöschten Titeln.
- config: Da pro Benutzer mehrere Clients möglich sind, werden auch die Informationen zu den Clients in der eigenen Datenbank gespeichert.
- log, statistics: Die Logdateien und Statistiken werden ebenfalls benutzerspezifisch gespeichert.

Bei der Darstellung dieser Daten durch die Webseiten gilt es nun, anhand des in userlog gespeicherten Cookies den angemeldeten Benutzer zu bestimmen und die entsprechende Datenbank auszuwählen. Dazu folgendes Beispiel:

Der Benutzer user1 verfügt über die individuelle Datenbank config_user1. Da vor dem Anzeigen jeder Seite das Skript check.php aufgerufen wird (Abs. 4.6.3), um zu überprüfen, ob das Cookie gültig ist, wäre es sinnvoll innerhalb dieses Skriptes die Variable $username zu definieren, welche auch im weiteren Ablauf gültig ist. Bei Zugriffen auf etwa die Beitrags-Datenbank wäre der Code zum Auswahl der Datenbank folgendermaßen abzuändern:

```
$whichdatabase = "config_".$username;
     mysql_select_db($whichdatabase, $clientstatus);
```

Damit würde abhängig vom ermittelten Benutzernamen auf die jeweilige Datenbank zugegriffen werden. Bei globalen Datensätzen (etwa Musiktitel) sind keine Veränderungen im Quelltext notwendig, da diese weiterhin in der zentralen Datenbank gesichert sind.

Sofern der Server über ausreichend Hauptspeicher verfügt, sollte sich diese Veränderung nicht drastisch auf die Performance auswirken. Zwar muss der *MySQL*-Server dadurch mehr Tabellen offen halten[161] als vorher, dafür werden aber Schreibzugriffe schneller durchgeführt, da *MySQL* standardmäßig bei INSERT

[161] http://dev.mysql.com/doc/mysql/en/Table_cache.html

Befehlen die komplette Tabelle sperrt (Lock)[162] – diese Locks verteilen sich nun auf verschiedene Tabellen.

Weiterhin erlaubt *MySQL* die Vergabe von sehr differenzierten Benutzerrechten. Für die Clients würden sich folgende Berechtigungen für den Zugriff auf den Server anbieten:

- main: Auf die Hauptdatenbank darf nur lesend zugegriffen werden (SELECT), die Benutzerinformationen sind komplett gesperrt.
- config_user1: Die Clients von etwa Benutzer user1 erhalten nur Zugriffsrechte auf die Datenbank config_user1, wodurch der Zugriff auf fremde Inhalte wirkungsvoll unterbunden wird.

Auf Seite der Clients sind fast keine Veränderungen notwendig. Lediglich die Anweisungen im Bereich „Programmelemente downloaden“ im *Updater.pl* Skript sowie die Statistikauswertung wäre umzuschreiben. Dabei kommt auf den Clients wie bisher nur eine Datenbank zum Einsatz, in welcher die Informationen aus main und config_user1 zusammengeführt werden.

5.4.3. Individuelle Nutzer-Verzeichnisse

Hat man eine Trennung der Benutzerdatenbanken realisiert, so ist es ein Leichtes, separate Benutzerverzeichnisse zu implementieren. Dabei wird, ähnlich wie bei der Auswahl der richtigen Datenbank, in die Seiten zur Übertragung von neuen Titeln das Benutzerverzeichnis zur Speicherung von Titeln hinzugefügt. Dies würde sich folgendermaßen darstellen:

```
$upload_dir = "/home/".$username. "/ads/";
```

[162] http://dev.mysql.com/doc/mysql/en/Table_locking.html

Hiermit würden Werbeblöcke für etwa Benutzer user1 in das Verzeichnis /home/user1/ads gespeichert werden.

Etwas problematisch an dieser Lösung ist, dass die eingesetzte *PureFTPd* Software ein so genanntes chroot auf das Benutzer-Verzeichnis durchführt. Dadurch kann ein Anwender nur auf Dateien und Verzeichnisse unterhalb seines Heimat-Verzeichnisses zugreifen – nicht aber mehr auf das Musikarchiv, welches ja im Verzeichnis /home/dira liegt.

Eine Lösung hierzu wäre, einen symbolischen Link im Heimatverzeichnis des Benutzers auf das Musikarchiv anzulegen. Zusätzlich muss *PureFTPd* mit der Option „—enable-symlinks“ kompiliert werden, damit es das Verfolgen von Links unterstützt. Dadurch wird das Musikarchiv innerhalb des Benutzerverzeichnisses eingeblendet.

Ein anderer Ansatz wäre die Verwendung von zwei verschiedenen Logins, wobei dies die Kommunikation zwischen Client und Server etwas aufblähen würde.

5.5. Audio-Bemusterung

Ein weiteres Einsatzgebiet für das DIRA-System ist die Nutzung als Bemusterungssystem für etwa Musikverlage. Diese befinden sich auf Grund der weit verbreiteten Kopierschutz-Verfahren auf Audio-CDs[163] in der Situation, dass sie die Bemusterungsexemplare separat ohne Kopierschutz herstellen müssen. Das liegt daran, dass kopiergeschützte CDs (welche nicht dem vorgeschriebenen CD-Standard entsprechen) sich nicht mehr in die digitalen Sendesysteme der Radiosender einspielen lassen.

[163] http://www.heise.de/ct/cd-register/

Durch einige Veränderungen am DIRA-System könnte man es in ein digitales Bemusterungssystem erweitern. Ein Bemusterungs-Set für einen Radiosender besteht unter anderem aus folgenden Elementen:

- Audio-CD: Hierbei handelt es sich meist um eine spezielle Edition ohne Kopierschutz.
- Informationsmaterial: Dabei handelt es sich um Pressetexte zur Geschichte des Künstlers, bevorstehende Auftritte oder Ähnliches.
- Bildmaterial: Da Radiosender meist über eine Webseite verfügen, benötigten sie zur zeitgemäßen Gestaltung dieser auch entsprechendes Bildmaterial.
- Audio-Promotionsmaterial: Dabei handelt es sich um kurze Sprachaufnahmen des Künstlers, die in das Programm eingebaut werden können (etwa: „Hallo, hier ist [Künstler] und ihr hört mein neues Album“).
- Merchandise: T-Shirts, Aufkleber, Taschen oder Ähnliches – dienen als mehr oder weniger offener Bestechungsversuch der Musikredakteure.

Diese Aufzählung stellt natürlich keine feste Vorgabe an eine Musik-Promotion dar, die Art der Bewerbung ist auch stark abhängig vom jeweiligen Sender. So wird wohl ein Klassiksender anders versorgt werden als ein Hit-Radio.

Abgeleitet aus der Zusammensetzung einer Promotions-Sendung ist der DIRA-Server in folgenden Punkten zu ändern:

- Textfelder: Damit den Radiosendern ausreichend Informationen über einen neuen Titel zur Verfügung gestellt werden, sollte der Bestand an Meta-Daten stark vergrößert werden.
- Bilddaten: Auf den Webseiten muss die Möglichkeit geschaffen werden, einem Titel mehrere Photos zuzuordnen.
- Umstrukturierung der Datenbank: Dabei gilt es abzubilden, dass einzelne Elemente zu einem übergeordneten Objekt gehören müssen. So muss zum Beispiel durch die Datenbank repräsentiert sein, dass ein Promotions-Titel zu

einem entsprechenden Musiktitel gehört – in der bisherigen Form benutzt das System nur eine flache Hierarchie, in der die Objekte nicht miteinander verknüpft sind.

- Vereinfachung der Webseiten: Da nun keine Werbeblöcke und Jingles geplant werden müssen, bietet sich eine Vereinfachung der Webseite an. Dabei sind die Elemente wie „Playlist verwalten“ zu entfernen, lediglich der Punkt „Neuen Titel hinzufügen“ muss entsprechend ausgebaut werden.

5.5.1. Hierarchische Datenbanken

Gerade für die beschriebene hierarchische Gliederung der Datensätze würde sich die Umstellung auf eine neuere Version von *MySQL* lohnen. Ab Version 4 wird ein neuer Datenbank-Typ namens InnoDB unterstützt, der unter anderem so genannte „Foreign Keys“ bietet. Wie der Name schon sagt, handelt es sich dabei um Indizes von Tabellen, welche aus einer anderen Tabelle entnommen sind. In Abbildung 5.5.1a wird deutlich, dass etwa die Tabelle „Texte“ abhängig vom jeweiligen Musiktitel ist. „Foreign Keys“ hätten nun den Vorteil, dass etwa beim Löschen eines Musiktitels automatisch auch die korrespondierenden Einträge in den entsprechenden Unter-Tabellen gelöscht werden. Dabei muss die Unter-Tabelle mit folgenden Anweisungen erstellt werden:

```
CREATE TABLE texte(parent_id INT, texte VARCHAR
    FOREIGN KEY (parent_id) REFERENCES musikarchiv(id)
            ON DELETE CASCADE) TYPE = INNODB;
```

Durch die “ON DELETE CASACDE”Anweisung sorgt die Datenbank dafür, dass, falls der Referenztitel aus der Tabelle musikarchiv gelöscht wird, der entsprechende Texteintrag auch getilgt wird[164]. Diese Verlagerung von Programmlogik in eingebaute Datenbankroutinen führt zu wesentlich kompakterem Code.

[164] http://dev.mysql.com/doc/mysql/en/InnoDB_foreign_key_constraints.html

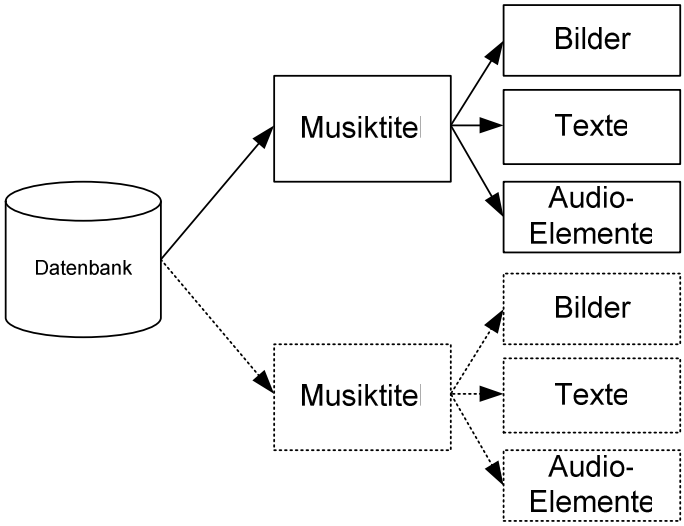

Abbildung 5.5.1a – Hierarchische Anordnung der Datenbank-Elemente

5.5.2. Weitere Überlegungen

Das System lässt sich leicht für die Übertragung von Bild- und Textmaterial anpassen. Interessant ist die Frage, auf welche Weise der Zugang zum transportierten Material auf Seite der Clients gewährleistet wird. Dabei sind zwei Lösungsansätze denkbar:

- Zugang über Webseiten: Dabei wird auf den Clients der *Apache*-Webserver installiert. Die Musikredakteure können über das lokale Netzwerk auf diesen Rechner zugreifen und erhalten die Informationen über eine vorgefertigte Webseite. Die Seite stellt dabei lediglich eine Vorlage dar, die eigentlichen Daten werden, wie beschrieben, über das Internet verteilt.
- Zugang über eine Netzwerkfreigabe: Bei dieser Lösung wird mittels des Samba-Servers[165] eine Netzwerkfreigabe erstellt, auf die von anderen Windows-Rechner zugegriffen werden kann. Hier erhält der Anwender keine vorgefertigte Oberfläche, sondern die Veröffentlichungen sind in einzelnen Verzeichnissen einsortiert. Darin befindet sich neben dem Audiotitel das

[165] http://de.samba.org/samba/samba.html

dazugehörige Promotionsmaterial in Form von Text- oder weiteren Audiodateien.

Der Vorteil der ersten Variante wäre, dass man die Informationen ansprechender aufbereiten kann. Will man allerdings eine Audio-Datei öffnen, so muss man diese zuerst auf die lokale Festplatte transferieren und speichern. Die zweite Lösung bietet dagegen direkten Zugriff auf die Dateien, bei entsprechender Nummerierung der Verzeichnisse lassen sich neue Titel auch leicht identifizieren.

Bei der Formatwahl für die Audio-Dateien sind folgende Punkte zu berücksichtigen:

- Qualität: Für die professionelle Nutzung durch Radiosender ist die Übertragung in einem nicht-verlustbehafteten Format sinnvoll. Dabei würde sich etwa der frei verfügbare *flac*-Codec[166] anbieten. Er bietet immerhin eine Kompressionsrate von etwa 50 % im Vergleich zu .WAV Dateien[167]. Die kodierten Dateien lassen sich dabei wieder bitgenau in das Original überführen.
- Digital Rights Management: Bei dieser zurzeit viel diskutierten Technik wird vorausgesetzt, dass eine Software zum Abspielen von Titeln über einen gültigen Schlüssel verfügt, der wiederum von einem Server bereitgestellt wird. Nur mit Hilfe dieses Schlüssels lässt sich eine Datei dekodieren, weiterhin kann zum Beispiel festgelegt werden, wie häufig diese abspielbar ist. Da aber in diesem Beispiel die Nutzung in digitaler Form gewünscht wird, kommt für die Anwendung als Bemusterungssystem eher der folgende Punkt in Betracht.
- Watermarking: Dabei werden in die Audiodaten unhörbare Informationen geschrieben, welche sich auch durch Neukodieren nicht mehr entfernen lassen. Auf diese Weise lässt sich zwar die Verbreitung von Musikstücken im Internet nicht verhindern, wohl aber kann man die undichte Stelle bei den Radiosendern ausfindig machen. Das Fraunhofer-Institut hat hierzu eine

[166] http://flac.sourceforge.net/

[167] http://flac.sourceforge.net/comparison.html

Technologie entwickelt, die aber leider nicht zur freien Verfügung steht[168]. Auch die deutschen Phonoverbände sind an einem Einsatz von digitalem Watermarking bei Musiktiteln interessiert.[169] Generell scheint es in der Open-Source Szene nur wenig Nachfrage nach einem solchen Verfahren zu geben – man findet nur ein entsprechendes Projekt auf Sourceforge[170], welches allerdings noch keine Veröffentlichung getätigt hat.

Gerade das Watermarking könnte für die Musikverlage ein entscheidendes Argument sein. Vor der Übertragung von neuen Titeln an die Radiostationen werden diese individuell per Wasserzeichen gekennzeichnet, was folgende Vorteile bringt:

- Automatisiertes On-Air Monitoring: Da die Wasserzeichen digital ausgelesen werden können, bietet sich hierbei die Möglichkeit der vollautomatischen Programmüberwachung – die Rechteinhaber können die von einem Sender gespielten Titel einfach anhand der darin kodierten Wasserzeichen identifizieren.
- Undichte Stellen: In Internet-Tauschbörsen findet man teilweise komplette Alben mehrere Wochen vor deren eigentlichem Erscheinungsdatum. Scheinbar haben zu viele Personen Zugang zu den Archiven der Musikredakteure, so dass hier Lecks in illegale Kanäle entstehen können. Diese kann man zwar mittels des Watermarkings nicht verhindern, wohl aber die Quelle identifizieren.

Vor allen in Zusammenhang mit dem beschriebenen Watermarking würde diese Lösung den Plattenfirmen helfen, zum einen Kosten durch Versand zu sparen, zum anderen, die Verbreitung von Titeln besser kontrollieren zu können.

---

[168] http://www.iis.fraunhofer.de/amm/download/watermark.pdf

[169] http://www.ifpi.de/recht/isrc.shtml

[170] http://sourceforge.net/projects/stegnu/

5.6. Szenarien für den Einsatz

Die Nutzung des DIRA-Systems für die zentral gesteuerte Beschallung von Einrichtungen erfordert relativ wenige Veränderungen an der bestehenden Software. Die Frage, wie viel Handlungsfreiraum man dem Anwender einräumt, lässt sich wohl nicht pauschal beantworten sondern ist stark abhängig vom Einsatzzweck. Im Folgenden sollen zwei unterschiedliche Szenarien mit verschiedenen Anforderungen durchgespielt werden.

5.6.1. Anforderungen an ein Campus Radio

In einem Gespräch mit Betreuern des Radio-100.000 Projektes des Fachbereichs wurde die Idee eines universitätsweiten Informations-Senders geboren. Dabei soll es den verschiedenen Fakultäten möglich sein, das Programm individuell zusammenzustellen. Da wohl keine Institution bereit ist, GEMA Gebühren zu entrichten, wird es sich bei dem Programm um ein reines Info-Radio ohne Musikbeiträge handeln. Die zu spielenden Wortbeiträge lassen sich dabei in zwei Kategorien einteilen:

- Universitätsweite Nachrichten: Vergleichbar mit der Startseite der Domain www.tu-berlin.de sollen hier Nachrichten eingespielt werden, welche für alle Studenten und Mitarbeiter relevant sind.
- Fakultätsbezogene Nachrichten: Die Information, dass etwa das Büro des Prüfungsausschusses der Maschinenbauer an einem bestimmten Termin geschlossen hat, ist wohl nur für einen Teil der Studenten von Bedeutung. Daher ist es sinnvoll, Nachrichten nach Bezug zu einer bestimmten Fakultät zu bündeln. Eventuell wäre sogar eine Unterscheidung nach Fachbereichen sinnvoll.

An diesem Punkt kommt die im vorherigen Abschnitt beschriebene Trennung der Benutzerprivilegien voll zum Einsatz Dabei würden die einzelnen Fakultäten ein Konto auf dem Zentralserver erhalten, unter welchem sie die Gestaltung des fakultätseigenen Senders vornehmen können. Zusätzlich zu den institutsbezogenen Beiträgen (die natürlich auch vom Institut eingespielt werden müssen) besteht die Möglichkeit, auf das Archiv der universitätsweiten Programmelemente zuzugreifen.

Der Vorteil dieser Lösung ist, dass sich damit ein ausgewogenes Informationsprogramm für die Zielgruppe (etwa die Studenten der Fakultät 1) zusammenstellen lässt. Ein Informationssender für alle Studenten der Universität dagegen hätte dagegen mit dem Problem zu kämpfen, dass entweder zu viele Nachrichten vorhanden sind (z.B. alle ausfallenden Sprechstunden der TU), welche in der Masse untergehen, oder aber, dass die Nachrichten zu allgemein sind und für den einzelnen Zuhörer nur einen geringen Informationswert haben.

5.6.2. Realisierung eines Campus Radios

Da der Erwerb einer Radio-Sendelizenz seitens der Universität mehr als unwahrscheinlich ist[171], sollte man die Wiedergabe des Programms über Streaming realisieren. Dabei wären folgende Anschaffungen notwendig:

- Pro Fakultät (oder Einrichtung), welche über einen eigenen Sender verfügen soll, ist ein Client-Rechner zu kaufen.
- Ein zentraler Server zur Speicherung aller anfallenden Beiträge.

Damit die Clients das Programm per Streaming zur Verfügung stellen können, bedarf es noch zweier zusätzlicher Software-Komponenten. Zum einen benötigt man einen Streaming-Server, zum anderen einen Encoder. Für den Streaming-Server

[171] [ALM2003], S. 429

bietet sich die Software *Icecast* an, eine andere Möglichkeit wäre Shoutcast, dieses Programm ist allerdings nicht unter der GPL veröffentlicht.

Der Encoder, der die Aufgabe hat, das zu streamende Material zu kodieren und an den Streaming-Server zu senden, könnte auf zwei Arten realisiert werden:

- Umprogrammierung des Feedme-Moduls: Wie bereits im TOX-System realisiert, besteht die Möglichkeit, die Encoder-Software *ices* über *Perl*-Skripten zu steuern. Dabei würden die zu spielenden Titel nicht mehr an *Jukepeg* zur Wiedergabe, sondern mittels *ices* an *Icecast* zum Streaming gesendet werden.
- Kodierung des analogen Audiosignals: Bei dieser Variante bleibt das DIRA-System unverändert, es kommt aber (neben dem *Icecast*-Server) zusätzlich die Software *LiveIce*[172] zum Einsatz. Diese ermöglicht es, die analoge Ausgabe der Soundkarte in das MP3-Format zu kodieren und an einen Streaming-Server zu senden.

Obwohl die zweite Variante etwas umständlicher ist (bestehende MP3-Dateien werden dekodiert, wiedergegeben und anschließend wieder nach MP3 kodiert), bietet sie zum einen den Vorteil, dass keine Änderungen an der DIRA-Software erfolgen müssen, zum anderen erlaubt sie, das Signal sowohl als analoges Audiosignal als auch als MP3-Stream wiederzugeben. In folgender Abbildung ist exemplarisch die Wiedergabe des Fachbereich 1 Radios dargestellt:

[172] http://star.arm.ac.uk/~spm/software/liveice.html

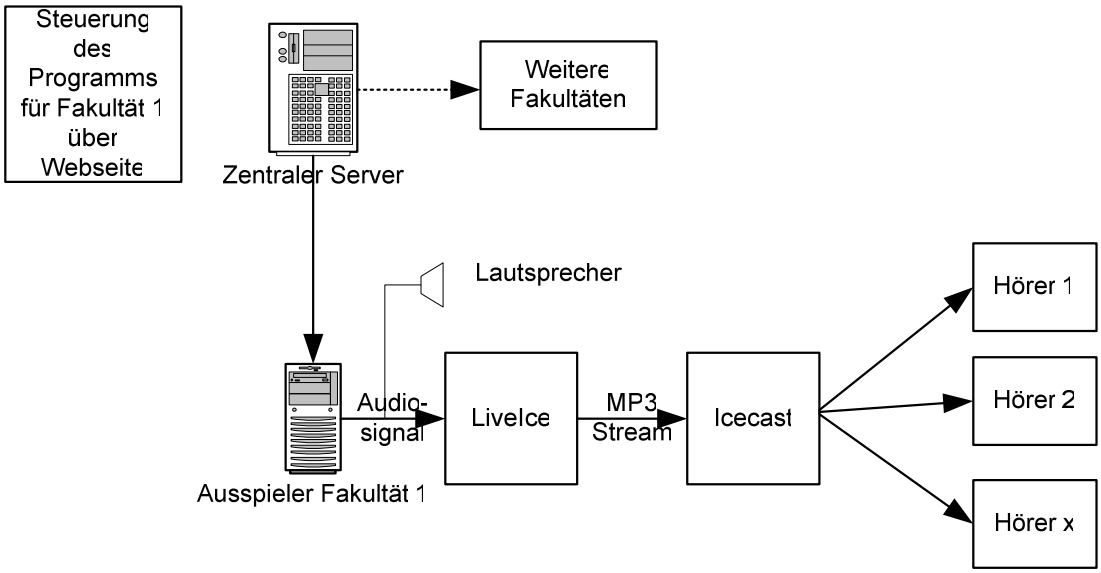

Abbildung 5.6.1 – Schema des Uni-Radios

Der Einsatz der Streaming-Technik in Verbindung mit dem DIRA-System mag widersprüchlich wirken, in diesem hier betrachteten Anwendungsszenario spricht aber eine Reihe von Punkten dafür:

- In der Regel wird ein Universitäts-Radio nicht ununterbrochen zur Beschallung von Räumen genutzt werden, sondern es soll einen zusätzlichen Informationskanal bieten. Da fast überall Rechner verfügbar sind und die Wiedergabe eines MP3-Streams selbst auf den ältesten Geräten gelingen sollte, bietet Streaming eine vernünftige Methode der Distribution.
- Da der Stream erst „vor Ort“ im Fachbereich erstellt wird, ergibt sich eine wesentlich geringere Netzlast als bei einem zentralem Ausspielrechner. Dabei werden die Titel nur einmal von Server zum Client übertragen, ansonsten fällt kein Traffik an. Gerade im Hinblick auf die Tatsache, dass die einzelnen Institute normalerweise über ein sehr gut ausgebautes internes Netzwerk verfügen, sollte die durch das Streaming entstehende Netzlast vertretbar sein.
- Dadurch, dass die jeweiligen Programme auf unterschiedlichen Rechnern erzeugt werden, erzielt man eine wesentlich höhere Betriebssicherheit. Zwar

ist es möglich, dass etwa durch einen Hardware-Defekt eine Fakultät zeitweise über kein Programm verfügt, die restlichen Einrichtungen wären davon aber nicht betroffen.

Von Seiten der Hardware wäre die Realisierung dieses Systems vermutlich ohne Probleme finanzierbar, problematischer dürfte die Erzeugung der benötigten Inhalte sein. Damit das Programm über längere Zeit interessant bleibt, müssten zumindest wöchentliche Aktualisierungen stattfinden – mit der eingehenden Frage, wer diese Tätigkeit finanziert.

Alternativ zum Einsatz an einer Universität wäre auch die Nutzung als Business-Radio denkbar[173]. Diese Systeme bedienen eine geschlossene Benutzergruppe innerhalb einer großen Firma. So könnte etwa ein großer Autokonzern Schulungssendungen für die Vertriebs-Mitarbeiter ausstrahlen, um diese besser auf Kundenfragen vorzubereiten. Die technische Realisierung eines solches Systems würde sich analog zum beschriebenen Uni-Radio gestalten – vermutlich nur mit umfangreicherer finanzieller Ausstattung.

5.6.3. Klassik-Abonnement-System

Da die Nutzung des DIRA-Systems für die Bemusterung von etwa Radiosendern bereits in Abs. 5.5 ausführlich beschrieben wurde, soll hier auf eine Variante dieser Anwendung eingegangen werden. Sie erfordert größere Veränderungen an der Client-Software, bietet dafür aber eine interessante Vermarktungsform im Endkundenbereich.

173

http://www.lfk.de/presseundpublikationen/publikationen/einzelpublikationen/Scherer/WorkshopSc000210.pdf

Die Anwendung lässt sich dabei als digitale Form des Musik-Abonnements beschreiben. Bei diesen Diensten zahlt der Benutzer eine feste Monatsgebühr und erhält dafür regelmäßig per Post eine CD mit aktuellen Titeln[174]. Durch Modifikationen am DIRA-System könnte man diesen Vorgang digital abbilden. Im Unterschied zu Anbietern wie Phonoline[175] oder iTunes[176] soll dabei kein Archiv geschaffen werden, bei dem der Anwender selber aus dem Sortiment des Betreibers auswählen kann, sondern ein Abonnement-Dienst, bei dem der Betreiber die Auswahl der Titel vorgibt.

Im Folgenden soll ein Szenario behandelt werden, bei dem der Anwender durch Abschluss eines Vertrages regelmäßig mit neuen Klassik-Aufnahmen versorgt wird. Dabei wird von folgenden Annahmen ausgegangen:

- Der Nutzer verfügt über einen Rechner mit Windows Betriebssystem
- Der Nutzer verfügt über eine schnelle Internetanbindung

Wegen der ersten Vorgabe ist die Entwicklung einer Windows-Client-Software notwendig, serverseitig sind sehr wenige Veränderungen erforderlich. Ein Anwender muss folgende Schritte zur Nutzung des Systems tätigen:

- Registrierung: Um den Dienst nutzen zu können, muss der Anwender sich auf einer Webseite registrieren – bei einem kommerziellen Dienst ist hierbei auch die Bankverbindung oder Ähnliches anzugeben. Ist dieser Vorgang abgeschlossen, erhält der Nutzer ein Passwort für seinen Zugang. Außerdem kann er seine gewünschte Musikrichtung angeben: Im Beispiel unseres Klassik-Abonnements wäre das etwa „Oper“, „Operette", „Barock“, „Leichte Klassik“, usw.
- Installation: Daraufhin muss von der Webseite die Clientsoftware transferiert werden. Dabei handelt es sich um eine ausführbare Windows-Setup Datei.

[174]http://www.timelife-europe.com/

[175] http://business.t-com.de/produkte/index.php?p_id=621

[176] http://www.apple.com/itunes/

Während der Installation wird dabei nach dem Benutzernamen und dem Passwort des Anwenders gefragt.

- Initialisierung: Da die Client-Software anfangs keinerlei Titel mit sich bringt, muss der Datenbestand über das Netz initialisiert werden. Hat der Benutzer sich etwa für „Barock“ entschieden, so etwa die letzten drei Veröffentlichungen des Dienstes als „Startguthaben“.

Sind diese Schritte erfüllt, so kann der Anwender die Software entweder manuell starten, um neu erschienene Titel zu übertragen, oder er lässt sie automatisch bei Systemstart laden, um so periodisch den Server nach Neuerscheinungen abzufragen. Im zweiten Fall wäre ein Symbol im System-Tray (der Schaltfläche links neben der Uhr) sinnvoll, welches etwa durch ein Ausrufezeichen das Vorhandensein neuer Titel anzeigt.

Aufgabe des Betreibers eines solchen Systems wäre die Bestückung der einzelnen Formate mit geeigneten Titeln. Für das Klassik-Abonnement wäre dabei sicherlich das Einspielen kompletter Aufnahmen sinnvoll. Die Abonnenten dagegen haben den Vorteil, dass ihr Archiv umso größer wird, je länger sie Kunde sind. Zusätzlich ist es möglich, Bonusmaterial wie Bilder, Interviews oder Ähnliches zusätzlich zu den Audiodaten zu übertragen, um so den Verlust des CD-Booklets zu kompensieren.

Die entscheidende Frage für das System ist, wie viele Rechte an den übertragenen Titeln man dem Anwender geben will. Die beiden Extreme sind in folgenden Punkten dargestellt:

- Bei der Installation erstellt die Software mit Hilfe des Passworts und der einmaligen Windows Security ID (SID)[177] einen Client-Schlüssel, welcher an den Server übertragen wird. Dieser verschlüsselt nun damit sämtliche an den Client übertragenen Titel. Der Client speichert die Musikstücke auf der Festplatte, die Wiedergabe ist nur mit der Client-Software möglich. Bevor ein Titel abgespielt wird, fragt der Client beim Server an, ob der Client-

[177] http://support.microsoft.com/default.aspx?scid=kb;en-us;314828

Schlüssel zum Titel passt und verweigert anderenfalls die Wiedergabe. Auch das Brennen auf CD muss mittels der Client-Software geschehen, hier kann der Betreiber festlegen, ob oder wie häufig dies geschehen darf. Kündigt der Benutzer sein Abonnement, so könnte der Betreiber dem Anwender die Abspielerlaubnis für seine Titel entziehen.

- Das andere Extrem wäre der komplette Verzicht auf alle DRM-Maßnahmen. Hierbei erhält der Anwender hochwertig kodierte Audiodateien, mit denen er – ähnlich einer CD - machen kann, was er will. Die Einwände der Musikindustrie, dass diese Dateien dann in Tauschbörsen getauscht werden können, sind wohl berechtigt - andererseits dürfte die Zielgruppe der Klassik-Abonnementen sich in anderen Internet-Kreisen bewegen als etwa 15-jährige.

Die wesentliche Frage ist, wie viele Einschränkungen die Verbraucher akzeptieren werden. Die Musikindustrie will schon heute in den Köpfen der Kunden verankern, dass der Erwerb einer CD nicht bedeutet, dass man diese in jedem CD-Spieler abspielen kann (siehe Abs. 2.4.1), der folgende Schritt wird wohl sein, Nutzungsbeschränkungen auf gekaufte Titel zu forcieren. Eine interessante Initiative zu diesem Thema wurde von der Zeitschrift c't gestartet, hier soll getestet werden, ob ein freiwillige Bezahlsystem für Musiktitel finanziell tragbar ist[178].

Für das Klassik-Abonnement ist wohl eine leichte Form des DRM sinnvoll. Wie in Abs. 5.5.2 beschrieben wird dabei in die Titel ein unhörbares Wasserzeichen eingebettet. In diesem Beispiel wäre etwa die Kundennummer des Abonnenten sinnvoll. Das verhindert zwar nicht die Weitergabe der Lieder, allerdings wird sich der Anwender genauer überlegen, ob er das Risiko eingehen will, als Quelle einer Raubkopie identifiziert zu werden.

Welche Rechte dem Nutzer letztendlich eingeräumt werden, hängt stark vom Verhandlungsgeschick des Betreibers mit der Musikindustrie ab. Vor allem in

[178] http://www.heise.de/ct/04/12/096/

Europa ist die Situation sehr unübersichtlich, da die Verwertungsrechte in verschiedenen Ländern oft bei unterschiedlichen Verlagen liegen.

5.7. Rechtliche Aspekte

Da das DIRA-System eine Plattform für eine Vielzahl von Anwendungen darstellt, ergeben sich daraus einige Fragen, welche nicht im Rahmen dieser Arbeit beantwortet werden können. Dabei handelt es sich vornehmlich um juristische Probleme, welche der Autor leider nur aus einer Laien-Perspektive betrachten kann.

5.7.1. GEMA-Gebühren

Die „Gesellschaft für musikalische Aufführungs- und mechanische Vervielfältigungsrechte“, kurz GEMA, hat die Aufgabe, die Rechte von Musikschaffenden zu wahren. Vereinfacht ausgedrückt bedeutet dies, dass ein Künstler sein neustes Werk bei der GEMA anmeldet und diese im Gegenzug für jede Aufführung, sei es im Radio, Fernsehen oder auf einer Dorfhochzeit, Lizenzgebühren erhebt. Die so erwirtschafteten Gebühren werden nach einem ausgehandelten Schlüssel an Verlage, Komponisten und Texter ausgeschüttet[179].

Man kann sich vorstellen, dass bei etwa 4,5 Millionen registrierten Titeln der Verwaltungsaufwand und der dazugehörige bürokratische Apparat entsprechend groß ausfallen. Zum Teil müssen sich Gerichte mit Lizenzforderungen der GEMA an eine Vogelart, welche eine registrierte Melodie zwitschert, beschäftigen[180]. Die Frage, ob für die Nutzung des DIRA-Systems GEMA-Gebühren zu entrichten sind, ließ sich nicht einwandfrei klären. Dabei gibt es zwei verschiedene Problemstellungen, welche von der Art des Einsatzes abhängen:

[179] http://www.gema.de/wirueberuns/

[180] http://www.legamedia.net/dy/articles/article_15853.php

- Aufführungsrechte: Ob für das Abspielen von Musik durch den DIRA-Client GEMA Gebühren entrichtet werden müssen, hängt davon ab, ob es sich um eine öffentliche oder eine private Aufführung handelt. Im Urheberrecht wird die Unterscheidung so definiert: "Die Wiedergabe eines Werkes ist öffentlich, wenn sie für eine Mehrzahl von Personen bestimmt ist, es sei denn, dass der Kreis dieser Personen bestimmt abgegrenzt ist und sie durch gegenseitige Beziehungen oder durch Beziehung zum Veranstalter persönlich untereinander verbunden sind[181]." Der Einsatz des Systems in einer Supermarktkette dürfte somit klar als öffentliche Aufführung gelten und muss daher bezahlt werden. Da diese Unternehmen aber in der Regel bereits über eine Beschallungsanlage verfügen, entrichten sie hierfür bereits eine GEMA-Pauschale, insofern ergibt sich für sie keine Veränderung.

  Bei der Nutzung als Distributionsplattform ist die Sachlage unklar, zwar stellt der Client den angeschlossenen Benutzern die Titel zur Verfügung, gibt diese aber nicht wieder. Dies geschieht erst, indem der Anwender die entsprechende Datei mit einem Abspielprogramm öffnet. Ob der Anwender nun für die Wiedergabe zahlen muss, oder auch der Betreiber für das Bereitstellen der Titel, ist unklar. Die GEMA hat leider nicht auf meine Anfrage diesbezüglich reagiert.
- Vervielfältigungsrechte: Der Name der GEMA beinhaltet den Begriff „mechanische Vervielfältigung“, was darauf hindeutet, dass sie bei der Herstellung von Tonträgern ebenfalls die Hand im Namen des Künstlers aufhält. Auf der Webseite finden sich keine Informationen bezüglich der nicht-mechanischen Vervielfältigung. Da das DIRA-System einmal eingespielte Titel auf mehrere weitere Rechner verteilt, kann man hier wohl von einer Vervielfältigung sprechen. Leider war nicht zu klären, ob dies unter die Kontrolle der GEMA fällt oder nicht.

[181] http://www.cfa-medien.de/vvor.htm

5.7.2. IFPI

Bei dieser Organisation handelt es sich um einen Dachverband der Tonträgerindustrie (International Federation of the Phonographic Industry)[182]. Sie kümmert sich um die Interessen der Phono-Industrie und dient als Ansprechpartner in allen diesbezüglichen Fragen. Dies ist im Zusammenhang mit dem DIRA-System dahingehend von Bedeutung, dass selbst wenn für die digitale Vervielfältigung keine GEMA-Gebühren zu entrichten sind, die Rechteinhaber die Vervielfältigung genehmigen müssen. Will man nun Musiktitel in das System einspielen, so kann je nach Anwendung entweder ein individueller Vertrag mit einem Musiklabel ausgehandelt werden oder eben ein genereller Vertrag mit der IFPI. Da die IFPI etwa 1000 Verlage vertritt, kann bei zweiter Lösung auf fast jeden verfügbaren Titel zugegriffen werden – anderenfalls nur auf Musikstücke eines bestimmten Hauses.

Auf Anfrage teilte die IFPI mit, dass es für die digitale Verbreitung von Musikstücken zwei verschiedene Abrechnungsmodelle gibt:

- Pauschalabrechnung: Dabei wird ein nicht näher genannter Prozentsatz des Jahresgewinns des Unternehmens an die IFPI abgeführt. Der Vorteil der Lösung ist, dass man dabei beliebig viele Kopien eines Titels erstellen kann. Der Verband behält sich allerdings eine technische Prüfung des Distributionssystems vor, um illegale Kopien zu unterbinden.
- Pro-Titel-Abrechnung: Hierbei muss festgestellt werden, wie viele Kopien eines Titels erstellt wurden. Die Datenträger müssen „inventarisiert und geschützt“ sein. Im Falle des DIRA-Systems wäre die Abrechnung relativ einfach. Dazu müsste man die Anzahl der Titel und der angeschlossenen Clients miteinander multiplizieren, um die Summe der getätigten Kopien zu berechnen.

Unklar in diesem Zusammenhang ist, ob die dadurch erlangten Rechte zur Distribution weltweit oder nur in Deutschland gelten. Zwar ist die IFPI eine

[182] http://www.ifpi.de

international tätige Organisation, die einzelnen Landesgruppen scheinen aber unabhängig voneinander zu agieren. Letztendlich müsste man, um eine klare Aussage bezüglich der Kosten zu bekommen, auch ein ausgearbeitet Nutzungskonzept vorlegen – diese Arbeit soll dafür aber lediglich als Grundgerüst dienen. Aus diesen Gründen ist leider eine einwandfreie Klärung von These 10 nicht möglich.

5.7.3. Sendelizenzen

Für den Betrieb eines Radiosenders wird in Deutschland eine Sendelizenz benötigt, zuständig für deren Vergabe sind die jeweiligen Landesmedienanstalten. Für den Bereich Internet-Radio gilt bislang noch die Entscheidung, dass auf „Grund der geringen Datenübertragungsrate oder -kapazität und der damit erreichbaren Klangqualitäten bzw. Bildgrößen keine mit herkömmlichen Rundfunk vergleichbare Suggestivkraft und Breitenwirkung erzielt werde“[183]. Daher gelten Internet-Radiosender bisher als Mediendienst und nicht als Rundfunk. Diese Erkenntnis stammt allerdings aus dem Jahr 1998, es bleibt abzuwarten, ob dies im Zuge des technischen Fortschrittes nicht eines Tages geändert wird.

Das DIRA-System, welches auf Grund seines Designs viele Schwächen des Internet-Radios umgeht, stellt in diesem Kontext wohl eine neue Form des Rundfunks dar und wurde dementsprechend auch gesetzlich noch nicht erfasst. Soll es als Beschallungssystem für den Handel eingesetzt werden, so ist zu überprüfen, ob dafür nicht eine Lizenz zu erwerben ist. Als Vergleich kann dabei der Service der Firma P.O.S- Radio aus Kiel[184] herangezogen werden. Diese Firma bietet ein Radioprogramm für den Einzelhandel an, welches über Satellit übertragen wird. Dabei hat der Kunde Auswahl aus mehreren vorgefertigten Formaten, die Werbezeitenplanung erfolgt dabei allerdings durch POS und gilt für alle Nutzer eines Formates. Für die Ausstrahlung eines solchen Programms musste bei der

[183] http://www.lfk.de/programme/neuedienste/internetrundfunk/main.html

[184] http://www.radio-pos.de/radio/index.php?seite=home&sprache=de

unabhängigen Landesmedienanstalt für Rundfunk und neue Medien eine Lizenz eingeholt werden[185].

In diesem Zusammenhang bleibt allerdings offen, ob dies auch für das DIRA-System gilt, da ja hier nicht über Satellit ein kontinuierlicher Datenstrom gesendet wird, sondern lediglich einzelne Titel übertragen werden, welche vor Ort zu einem Radioprogramm zusammengesetzt werden. Dazu ließ sich lediglich ein Dokument der Landesanstalt für Kommunikation in Baden-Württemberg finden, welches klarstellt, dass so genanntes Point-Of-Sale Radio sowohl „[…] unter den Rundfunkstaatsvertrag als auch unter den Mediendienste-Staatsvertrag fallen [kann][186]“.

5.8. Zusammenfassung

Die entwickelte Software stellt neben der erzielten Funktionalität einen Beitrag zur Grundlagenforschung auf dem Gebiet der verteilten Radioautomation dar. Sie läuft robust und bietet einen an die erweiterten Fähigkeiten des Systems angepassten Funktionsumfang.

Anhand der beschriebenen Szenarien wird deutlich, dass das DIRA-System als Plattform für eine ganze Reihe von unterschiedlichen Anwendungen dienen kann. Die Bandbreite reicht dabei von Netzwerklast reduzierenden Distributionsmethoden bis hin zu Abonnement-Systemen für digitalisierte Musik. Dabei stellt die offensichtlichste Einschränkung, der Verzicht auf den Live-Betrieb, für die meisten Anwendungen kein großes Problem dar. Speziell die Distribution von Musikstücken durch die Verlage an festgelegte Empfänger (z.B. Radiostationen) scheint ein aussichtsreicher Einsatzzweck zu sein.

---

[185] http://www.ulr.de/ULR_Akt_Presse03/einzelausgabe_wai_12.shtml

[186] http://www.lfk.de/presseundpublikationen/publikationen/einzelpublikationen/Scherer/WorkshopSc000210.pdf

In diesem Fall sind auch die rechtlichen Fragen geklärt, da ja der Initiator der Distribution auch über die benötigten Rechte verfügt. Für alle Anwendungen, bei denen durch eine 3. Partei fremde Werke benutzt werden, gestaltet sich die Situation schwieriger. Hier ist durch die vielen verschiedenen nationalen Musikverlage die Abdeckung ganz Europas mit erheblich größerem Aufwand verbunden als dies etwa in den Vereinigten Staaten der Fall wäre. Diese juristischen Implikationen können aber nicht in dieser Arbeit behandelt werden. Es bedarf vermutlich einer größeren finanziellen Ausstattung, um die notwendigen Rechte zu erhalten.

Im Bezug auf die aufgestellten Thesen konnte durch die Entwicklung der DIRA-Software als Proof-of-Concept gezeigt werden, dass sowohl individuelle Anforderungen für den Radiobetrieb mit Internettechniken realisierbar sind (These 1) als auch eine Übereinstimmung der Meta- und Audiodaten durch vorhandene Netzwerkprotokolle sichergestellt werden kann (These 2). Ebenfalls konnten durch den kombinierten Einsatz von Multicast, Pull-Technologien, Peer-to-peer Übertragungen und Distributed Computing die Probleme bei der Übertragung von Audio und/oder Bildmaterial umgangen werden (These 4) und so eine Distribution bei weitgehenden Verzicht auf Streaming realisiert werden (These 3). Die für das System benutzte Hardware besteht komplett aus Low-Cost Komponenten, so dass These 5 ebenfalls als bestätigt gelten kann.

Durch die in der Simulation gewonnen Daten, sowie durch theoretische Überlegungen wurde gezeigt, dass die Datenmenge und Last auf dem Server durch den Einsatz von Pull-Technologien erheblich sinkt (These 6). Eine weitere Reduzierung ist durch den Einsatz von Peer-to-Peer-Transfers zu erwarten, dies muss aber noch in einer Simulation und im Praxistest bestätigt werden (These 7). Dasselbe gilt für die Erhöhung der Betriebsstabilität durch den Einsatz von verteilten Rechnerstrukturen (These 8), dies konnte auf Grund der limitierten Finanzmittel lediglich theoretisch festgestellt werden.

Lediglich die rechtlichen Aspekte bei der Vervielfältigung von Musiktiteln über das Internet sind weiterhin unklar. Dabei ist zu bedauern, dass keine der zu diesem Thema angeschriebenen Firmen auch nur geantwortet hat. Somit bleibt die Frage, ob

das DIRA-System sich auf eine juristisch solide Grundlage stellen lässt, vorerst leider offen (These 9).

6. Fazit

Durch die Erweiterung des Internet-Streaming-Systems TOX um die beschriebenen Technologien konnte diese Anwendung in entscheidenden Punkten verbessert werden. Dabei sind vor allem die sehr gute Skalierbarkeit der Datenübertragung sowie die Robustheit bei der Wiedergabe hervorzuheben. Als äußerst positiv ist in diesem Zusammenhang außerdem die Stabilität der Client-Software zu bewerten. Diese lief über mehrere Wochen ununterbrochen durch, ohne dass es dabei zu nennenswerten Fehlern gekommen wäre. Dies mag zum Teil darin begründet sein, dass der Server über eine recht unzuverlässige Netzwerkanbindung verfügt und daher bei den Transferroutinen besondere Sorgfalt auf die Fehlerüberprüfung gelegt wurde.

In der derzeitigen Form eignet sich das DIRA-System als Plattform für eine Reihe von unterschiedlichen Anwendungen. Der Einsatzbereich ist dabei zwischen automatischem Distributionssystem und Radio-Ausstrahlung angesiedelt. Vor allen durch die erhebliche Reduzierung des Transfervolumens ist neben den beschriebenen Anwendungen auch eine Nutzung im wesentlich größeren Rahmen denkbar. Im Unterhaltungs-Bereich ist seit einiger Zeit ein starker Trend zum PC als Medienzentrale im Wohnzimmer feststellbar[187]. Dabei konvergieren klassische Unterhaltungsmedien wie Fernsehen oder Radio mit internetbasierten Medien wie Webradio oder Anwendungen wie elektronischen Programmplanern. Ein Großteil der Video-Inhalte wird immer noch per Funk-Übertragung (sei es über Antenne oder Satellit) empfangen, was dem propagierten individuell zusammenstellbaren Programm abträglich ist.

[187] http://www.video-magazin.de/d/23998

In diesem Zusammenhang bietet das DIRA-System die Grundlage für eine Plattform zur intelligenten Distribution von Inhalten über das Internet. Dabei wäre vorstellbar, dass die Medienzentrale anhand der vorgegebenen Wünsche der Nutzer die Inhalte entweder per Internet durch die beschriebenen Übertragungsmethoden (Multicast, Peer-to-peer etc.) oder auf herkömmliche Weise per Funk oder Kabel empfängt, lokal speichert und erst dann wiedergibt, wenn dies gewünscht wird.

Solch ein Systemdesign setzt allerdings wesentlich komplexere Hard- und Software voraus, vor allem im Hinblick auf die Sicherung der empfangenen Elemente gegen unerlaubte Vervielfältigung werden die Medienkonzerne umfangreiche Schutzmaßnahmen fordern, ehe sie ihre Inhalte zur Verfügung stellen.

Dieses recht visionäre Szenario für das DIRA-System zeigt aber, dass die im Rahmen dieser Arbeit gewonnen Erkenntnisse durchaus auf zukünftige Massentechnologien anwendbar sind. Eine sehr hohe Betriebsstabilität, sowie für den Betreiber niedrige Betriebskosten sind für den Erfolg eines Nachfolgesystems zum Fernsehen unabdingbar.

Die meisten der für diese Anwendung notwendigen Routinen sind bereits in der DIRA-Software vorhanden, so dass dieses System als Grundlage für weitere Untersuchungen auf diesem Gebiet dienen kann. Dabei würde sich unter anderem der verbesserte Schutz der Inhalte, automatische Peer-to-peer-Transfers oder die Klärung der juristischen Aspekte zur Vertiefung anbieten.

Im theoretischen Teil der Arbeit konnte gezeigt werden, dass sich durch die Kombination verschiedener Übertragungstechniken etliche Schwachstellen bestehender Internettechnologien in diesem Feld ausgleichen lassen. Weiterhin wurden die durch die verteilte Architektur dieses Systems entstehenden Probleme analysiert und, falls möglich, behoben.

Durch die Kombination aus theoretischer Analyse sowie praktischer Umsetzung eines Proof-of-Concept-Systems ließ sich durch den Einsatz von Multicast, Pull-Technologien, Peer-to-peer-Übertragungen und Distributed Computing eine

nachhaltige Verbesserung von Audio- und Videoanwendungen im Internet nachweisen.

7. Ausblick

Um das System einer größeren Benutzergruppe zugänglich zu machen, lässt sich die Entwicklung einer Client-Software für das Windows Betriebssystem nicht umgehen. Die Installation der Datenbank sowie die Bedienung der Skripte erfordern zu viel Fachwissen, so dass hiermit keine größere Durchdringung erzielt werden kann.

Optimal wäre in diesem Zusammenhang ein Client-Dämon, der in die Tray-Leiste (das Feld im rechten, unteren Bildschirmbereich) verkleinert werden kann und so dem Anwender gar nicht auffällt. Dieser führt, wie beschrieben, periodisch Updates durch und spielt außerdem die vorhandenen oder neuen Titel ab.

Da die DIRA-Software mit Erscheinen der Arbeit unter der General Public License frei verfügbar sein wird, steht es jedem Interessiertem offen, diese für eigene Einsatzzwecke zu nutzen. Durch die Veröffentlichung der Quelltexte erhoffe ich mir, Gleichgesinnte zu finden, die die Entwicklung des Systems auch nach Fertigstellung dieser Arbeit weiter vorantreiben.

8. Anhang

8.1. Quelltexte

Das Archiv mit den Quelltexten sowohl für die Webseiten als auch die Perl-Skripten kann unter der URL http://www.tektribe.de/diss/sources.tar.bz2 abgerufen werden.

8.2. Verzeichnis der Formeln

8.3. Verzeichnis der Abkürzungen

CD	Compact Disc
DAT	Digital Audio Tape
MP3	MPEG-Audio Layer 3
PCM	Pulse Code Modulation
PC	Personal Computer
EFM	Eight-to-fourteen Modulierung
CIRC	Cross-Interleaved Reed Salomon Coding
SCMS	Serial Copy Management System
CD-R	Compact Disc-Recordable
CD-MO	Magneto-optische CD
ATRAC	Adaptive Transform Acoustic Coding
DCC	Digital Compact Cassette
SDAT	Serial Digital Audio Tape

MPEG	Motion Picture Expert Group
WMA	Windows Media Audio
CD-ROM	Compact Disc – Read Only Memory
AAC	Advanced Audio Coding
DVD	Digital Versatile Disc
USB	Universal Serial Bus
DAB	Digital Audio Broadcast
GEMA	Gesellschaft für musikalische Aufführungs- und mechanische Vervielfältigungsrechte
SQL	Structured Query Language
RAID	Redundand Array of Independent Discs
ARPA	Advanced Research and Planning Agency
ISDN	Integrated Services Digital Network
TCP	Transmission Control Protocol
IP	Internet Protocol
IETF	Internet Engineering Task Force
RSVP	Reservation Procotol
MBONE	Multimedia Backbone
IGMP	Internet Group Message Protocol
DIRA	Distributed Radio System
TOX	Total Access System
HE	Höheneinheit
DSL	Digital Subscriber Line
GPL	General Public License
HTML	Hyptertext Markup Language
CGI	Common Gateway Interface
ASP	Active Server Pages
JSP	Java Server Pages
PHP	Personal Hypertext Processor
ANSI	American National Standards Institute
FTP	File Transfer Protocol
PERL	Practical Extraction and Reporting Language
ACL	Access Control List

TLS	Transport Level Security
SCSI	Small Computer Storage Interface
CHROOT	Changed Root
DRM	Digital Rights Management
IFPI	International Federation of the Phonographic Industry

8.4. Verzeichnis der Abbildungen

8.5. Verzeichnis der Tabellen

8.6. Verzeichnis der benutzten Webseiten

http://atknoll1.informatik.tu-tuenchen.de/Zope/tum6/lectures/seminars/ws0203/avformate/bierbaum.pdf

http://bitconjurer.org/BitTorrent/

http://business.t-com.de/produkte/index.php?p_id=621

http://c2.com/cgi/wiki?SinglePointOfFailure

http://cocoon.apache.org

http://come.to/dcc-faq

http://dance.kunden-efactory.de/statistik/stats.php

http://de.php.net/features.file-upload

http://de.samba.org/samba/samba.html

http://de2.php.net/features.safe-mode

http://dev.mysql.com/doc/mysql/en/InnoDB_foreign_key_constraints.html

http://dev.mysql.com/doc/mysql/en/Replication.html

http://dev.mysql.com/doc/mysql/en/Table_cache.html

http://dev.mysql.com/doc/mysql/en/Table_locking.html

http://en.wikipedia.org/wiki/Gnutella

http://flac.sourceforge.net/

http://flac.sourceforge.net/comparison.html

http://home.t-online.de/home/richard.buechter/theorie.html

http://lftp.yar.ru/

http://linux-ha.org/

http://mail.phys-iasi.ro/Library/Computing/mpeg_layer_3.htm

http://mp3gain.sourceforge.net/

http://news.netcraft.com/archives/web_server_survey.html

http://selfhtml.teamone.de/diverses/htaccess.htm

http://setiathome.ssl.berkeley.edu

http://setiathome.ssl.berkeley.edu/cacm/cacm.html

http://slacksite.com/other/ftp.html

http://sql.idv.edu/thema/dbgrundlagen/begriffsklaerung.htm

http://star.arm.ac.uk/~spm/software/liveice.html

http://support.microsoft.com/default.aspx?scid=kb;en-us;314828

http://web.utk.edu/~smarcus/History.html
http://www.aes.org/technical/documents/i2.pdf
http://www.ahinc.com/raid.htm
http://www.airwindows.com/encoders
http://www.amazon.de
http://www.apple.com/ipod/
http://www.apple.com/itunes/
http://www.atmel.com/dyn/resources/prod_documents/doc4109.pdf
http://www.bio-itworld.com/news/111703_report3789.html
http://www.bitbox.co.uk/jukepeg/
http://www.cert.org
http://www.cfa-medien.de/vvor.htm
http://www.cpan.org
http://www.debian.org
http://www.redhat.de
http://www.debian.org/security/2003/dsa-403
http://www.drbd.org/
http://www.ebay.de
http://www.edonkey2000.com/
http://www.extremetech.com/print_article/0,3428,a=25002,00.asp
http://www.fh-wedel.de/~si/seminare/ss02/Ausarbeitung/9.digitalaudio
http://www.fokus.gmd.de/research/cc/cats/products/Flyer-Kommandoanlage.pdf
http://www.freeamp.org
http://www.gadegast.de/frank/mpegfaq/mpe632.html
http://www.gema.de/
http://www.gema.de/wirueberuns/
http://www.geotrusteurope.com/de/
http://www.google.com/press/funfacts.html
http://www.cushat.co.uk/prime/
http://www.heise.de/ct/04/12/096/
http://www.heise.de/ct/cd-register/
http://www.heise.de/ct/cd-register/default.shtml
http://www.ibm.com/developerworks/java/library/j-p2p

http://www.icculus.org/mp3check/mp3check
http://www.icecast.org
http://www.ietf.org/html.charters/pim-charter.html
http://www.ietf.org/internet-drafts/draft-murray-auth-ftp-ssl-13.txt
http://www.ifi.unizh.ch/mml/publications/diplomarbeiten/schrieber.pdf
http://www.ifpi.de
http://www.ifpi.de/jb/2003/32-37.pdf
http://www.ifpi.de/jb/2003/32-37.pdf
http://www.ifpi.de/recht/isrc.shtml
http://www.iis.fraunhofer.de/amm/download/watermark.pdf
http://www.iis.fraunhofer.de/amm/techinf/layer3
http://www.kazaa.com/us/index.htm
http://www.kgw.tu-berlin.de/~cbradter/labor2001/Band.pdf
http://www.lanl.gov/
http://www.legamedia.net/dy/articles/article_15853.php
http://www.level3.com/2577.html
http://www.lfk.de/presseundpublikationen/publikationen/einzelpublikationen/Scherer/WorkshopSc000210.pdf
http://www.lfk.de/presseundpublikationen/publikationen/einzelpublikationen/Scherer/WorkshopSc000210.pdf
http://www.lfk.de/programme/neuedienste/internetrundfunk/main.html
http://www.linuxvirtualserver.org/
http://www.linuxvirtualserver.org/HighAvailability.html
http://www.minidisc.org/faq_sec_4.html
http://www.modatic.net/audio/mp3_encoder_comparison.php
http://www.modsecurity.org/
http://www.mp3dev.org/mp3/
http://www.muenster.de/~asshoff/physik/cd/cdplayer.htm
http://www.mysql.com/doc/de/Indexes.html
http://www.oneoffcd.com/info/historycd.cfm
http://www.openp2p.com/pub/a/p2p/2001/12/14/topologies_one.html
http://www.openp2p.com/pub/a/p2p/2002/01/08/p2p_topologies_pt2.html
http://www.oto-online.com/june00/foc_mini.html

http://www.physik.uni-bielefeld.de/~hempel/diplom/node50.html
http://www.proftpd.net/
http://www.ptb.de/de/org/q/q4/q42/ntp/ntp_main.htm
http://www.pureftpd.org/
http://www.pureftpd.org/README.TLS
http://www.radiomanager.de/Formate.htm
http://www.radio-pos.de/radio/index.php?seite=home&sprache=de
http://www.radioszene.de/chr.htm
http://www.rarsoft.com
http://www.rcseurope.de/produkte/home.html
http://www.rms.de/order_check/download/angebote/radioformate.pdf
http://www.rootforum.de/forum/viewtopic.php?t=19898
http://www.shoutcast.com
http://www.sims.berkeley.edu/courses/is224/s99/GroupG/report1.html
http://www.stunnel.org
http://www.suphp.org/Home.html
http://www.suse.de
http://www.technews.vt.edu/Archives/2003/Nov/03665.htm
http://www.tektribe.de/magisterarbeit.pdf
http://www.timelife-europe.com/
http://www.top500.org/list/2003/11/
http://www.tu-
chemnitz.de/informatik/RA/kompendium/vortraege_96/CDROM/cdrom0.html
http://www.tvhandbook.com/support/pdf_files/audio/Chapter8_1.pdf
http://www.typo3.net
http://www.ulr.de/ULR_Akt_Presse03/einzelausgabe_wai_12.shtml
http://www.vergenet.net/linux/fake/
http://www.verisign.com/products/site/index.html
http://www.warptec.com
http://www.webopedia.com/TERM/S/Skript_kiddie.html
http://www.welt.de/data/2003/08/15/153186.html
http://www.winzip.com
http://www.wps.com/FidoNet

http://www.wu-ftpd.org/

http://www.zend.com/store/products/zend-engine.php

http://www.zenon-media.com/

http://www-user.tu-chemnitz.de/~mfie/compproj/index.htm

http://yp.icecast.com

8.7. Literaturverzeichnis

[ABK1994] Abel, Karl-Dietrich (Hrsg), Telemedien für morgen; Vistas Verlag, 1994

[ALM2003] Arbeitsgemeinschaft der Landesmedienanstalten in der Bundesrepublik, Privater Rundfunk in Deutschland – Jahrbuch der Landesmedienanstalten; Vistas Verlag, 2003

[ANO1999] Anonymous, Sicherheit im Internet und im lokalen Netz; Markt und Technik Verlag, 1999

[BAS2000] Baloui, Said; Access 2000 Kompendium - Datenbanken planen, entwickeln, optimieren; Markt und Technik Verlag, 2000

[COD2000] Comer, Douglas, Internetworking with TCP/IP; Prentice Hall Verlag, 2000

[COG2000] Coulouris, George / Dollimore, Jean / Kindberg, Tim, Distributed Systems, Addison-Wesley Verlag, 2000

[COJ1997] Conrad, Jan-Friedrich, Musik-Elektronik; PPV Verlag, 1997

[EBU1998] Laven, P.A. (Hrsg), EBU / SMPTE Task Force for Harmonized Standards for the Exchange of Programme Material as Bitstreams – Final Report and Results August 1998; European Broadcasting Union (EBU), 1998

[GRA2000] Grothe, Andreas, Verflüchtigt: Der Zahn der Zeit nagt an digitalen Daten, c't 24/2000, Heinz Heise Verlag 2000

[HAS2002] Hansen, Sven; Unerhört gut! - MP3-Nachfolger im c´t-Hörtest; c't 19/02; Heinz Heise Verlag 2002

[INW1991] Inmon, William, Client-Server Anwendungen – Planung und Entwicklung; Springer Verlag, 1994

[KAS2003] Karzauninkat, Stefan; Google zugemüllt Spam überschwemmt die Suchergebnisse; c't 20/03, Heinz Heise Verlag 2003

[KOM1998] Kofler, Michael, Linux – Installation, Konfiguration, Anwendung; Addison-Wesley Verlag, 1998

[LAP1995] Lala, Parag, Fault tolerant and fault testable hardware design; Prentice Hall Verlag, 1995

[LAT1995] Lauer, Thomas, Internet – Kompendium; Markt & Technik Verlag, 1999

[MCR1994] McLeish, Robert, Radio Production – Manual für Broadcasters; Focal Press Verlag, 1994

[PEL2000] Peterson, Larry / Davie, Bruce, Computernetze – Ein modernes Lehrbuch; dpunkt Verlag, 2000

[POM2001] Postel, Matthias, Internet und Fernsehen – Vom asynchronen zum synchronen Content bei Live-Events; LIT Verlag, 2001

[SCE2000] Schmid, Egon (Hrsg.); PHP Handbuch, PHP Documentation Group, 2000

[SCW1995] Schiffner, Wolfgang, Lexikon Tontechnik; Bärenreiter Verlag, 1995

[SIA2001] Stiller, Andreas; Fröhliche Oldies - Der PC feiert seinen 20sten Geburtstag; c't 16/01; Heinz Heise Verlag 2001

[SIE1998] Siever, Ellen; Spainhour, Stephen; Padwardhan, Nathan; Perl in a nutshell; O'Reilly Verlag, 1998

[SOM2002] Soltau, Michael, Unix/Linux Hochverfügbarkeit; mitp Verlag, 2002

[VRA2002] Vrenios, Alex, Linux Cluster Architecture; Sams Verlag, 2002

[WAJ2001] Watkinson, John, The MPEG-Handbook, Focal Press, 2001

[WAL2001] Wall, Larry / Christiansen, Tom / Orwant, Jon, Programmieren mit Perl; O'Relly Verlag, 2001

[WEH1989] Werle, Horst (Hrsg), Technik des Rundfunk, Technik der Systeme, Rundfunkversorgung; R. v. Decker Verlag, 1989

[WIG1977] Wiederhold, Gio, Database Design; McGrawe-Hill, 1977

[ZOA1996] Zomoya, Albert (Hrsg), Parallel and distributed computing handbook; McGrawe-Hill Verlag, 1996

[ZOV2000] Zota, Volker / Buschmann,Andree, Konkurrierende Klangkonserven - Verlustbehaftete Audioformate im Vergleich; c't 23/00, Heinz Heise Verlag, 2000