Hochschule für Technik, Wirtschaft und Kultur Leipzig (FH)

Fachbereich Informatik, Mathematik und Naturwissenschaften

Lehrbereich Informatik

Interaktionskomponenten und wechselseitige Abstimmung der Ressourcennutzung in einem Distance-Learning-System

Magisterarbeit

Zur Erlangung des akademischen Grades

Magister scientiarum (M.Sc.)

Leipzig, 14. April 2003

vorgelegt von

Rebs, Andreas
geb. am: 11.04.1972

Studiengang Informatik

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Quelltextverzeichnis

Kapitel 1

Einleitung

In den letzten Jahren hat sich das Anwendungsspektrum auf dem Gebiet des multimedialen Informationsaustausches nicht zuletzt wegen leistungsfähigerer Rechentechnik und neuer Netzwerk-Technologien im Heimanwender- und Business-Bereich, wie etwa T-DSL, enorm erweitert. Gerade durch die Bereitstellung immer größer werdender Bandbreiten für Anbindungen von Personal-Computern an das Internet wurde der Einsatz von Technologien zur Übertragung von audiovisuellen Informationen interessant, da diese mit einem hohen Datenaufkommen verbundene Art des Informationsaustausches der zwischenmenschlichen Kommunikation nahe kommt und somit einen breiten Anwendungsbereich in sich birgt.

Ein hierbei immer wieder diskutiertes Anwendungsgebiet für die Übertragung audiovisueller Informationen stellt das sogenannte Distance-Learning über das Internet dar. Hierunter versteht man eine entscheidende Weiterentwicklung traditioneller Formen des Fernunterrichts oder Fernstudiums, wobei Lehrender und Lernende weder am gleichen Ort noch zur gleichen Zeit mit dem Lernen beschäftigt sein müssen. Distance-Learning eignet sich somit sowohl zur flexiblen Weiterbildung nach Bedarf als auch für kursorientierte Lehrveranstaltungen jeglicher Art und ermöglicht durch den audiovisuellen Informationsaustausch eine Lernumgebung, welche viele Einschränkungen des traditionellen Fernunterrichtes der Vergangenheit angehören lässt. Durch den Einsatz von Interaktionsdiensten unterschiedlichster Form wird es Lernenden ermöglicht, beispielsweise aktuell präsentierten Lernstoff zum besseren Verständnis nach Belieben zu diskutieren oder sonstiges Wissen hierüber auszutauschen, und das in Echtzeit [STE00].

Mittlerweile wurden eine ganze Reihe solcher Distance-Learning-Systeme entwickelt. Hierzu gehören unter anderem Systeme wie JaTeK [FRI99], KBS Virtual Classroom [FRI99], WINFOLINE [KIV03] und OVID. Letzteres wird in [KUR00], [ROS00] und [REN00] ausführlich beschrieben. Es ist für die Betriebssystem-Umgebung von Microsoft Windows^{TM}ausgelegt und beinhaltet neben einem Modul zur Präsentation des Lernstoffes auch zusätzliche Module zur Kommunikation und zum Datenaustausch der Lernenden und Lehrenden untereinander.

Hierzu gehören sowohl die Zusatzdienste zur Benutzerkommunikation via Chat, Whiteboard, Audio/Video-Konferenz und eMail als auch ein integrierter Dienst zum Datentransfer via FTP. Außerdem berücksichtigt OVID auch die für Systeme dieser Art üblichen Anforderungen und Techniken bezüglich der Verwaltung von Nutzern und Diensten.

Im Rahmen der hier vorliegenden Arbeit werden auf Grundlage des angesprochenen Distance-Learning-Systems OVID weitergehende Betrachtungen angestellt, welche zum einen alternative Konzeptionen und Implementierungen für die in OVID integrierten Interaktionskomponenten aufzeigen und zum anderen verschiedene Technologien zur Präsentation von Lehrinhalten innerhalb des Distance-Learning-Systems vorstellen und deren eventuell vorhandene Vor- und Nachteile hinterfragen. Bei den hier angesprochenen Technologien zur Präsentation des Lernstoffes ist das Hauptaugenmerk dabei vornehmlich auf die Integration der für virtuelle Vorlesungen typischen Video-Streams innerhalb der Lehrstoff-Präsentationen ausgerichtet.

Eine alternative Konzeption von Interaktionskomponenten macht sich insbesondere deshalb erforderlich, weil beispielsweise der im OVID integrierte Dienst zur Audio- und Videokommunikation nur unter Verwendung spezieller Hardware zur Verfügung steht. Hierfür gilt es also ein Konzept zu entwickeln, welches eine derartige Kommunikation unter derzeitigen Gesichtspunkten auf nahezu jedem Rechner ermöglicht, wobei aber auch hier einige Mindestanforderungen - insbesondere die verfügbare Bandbreite und Rechenleistung betreffend - unumgänglich sein werden. Auch die Nutzung des integrierten Dienstes zur Präsentation von Lehrinhalten ist im System OVID nur bei einer Netzanbindung über ein LAN in vollem Umfang möglich, da die in diesem Dienst vorkommenden Audio/Video-Inhalte entsprechend des Konzeptes vom OVID eine Bandbreite von mindestens 1.2 Mbit/s [ROS00] voraussetzen und somit infolge des damit verbundenen Datenaufkommens eine Nutzung dieses Dienstes im Heimbereich verhindern.

Die Auswertung und Abstimmung der vorhandenen Systemressourcen für das in dieser Arbeit betrachtete Distance-Learning-System spielt in Bezug auf die Verfügbarkeit der Zusatzdienste eine entscheidende Rolle. Die sich hieraus ergebenden Anforderungen und Möglichkeiten zur Umsetzung werden an entsprechender Stelle ebenfalls aufgezeigt.

Der Umstand, dass bei dem hier entwickelten System GWOTS die Zusatzdienste für die Nutzerkommunikation via Chat, Whiteboard und Videokonferenz keine eigenständigen Anwendungen darstellen, bringt auch einige grundlegende Unterschiede gegenüber der im OVID vorhandenen Kommunikationstruktur zwischen Dienstanbieter und Dienstnutzer mit sich. Somit werden auch die hierfür vorgenommenen Anpassungen Gegenstand der nachfolgenden Ausführungen sein und an gegebener Stelle ausführlich diskutiert.

Die vorliegende Arbeit gliedert sich in 6 Kapitel, deren Inhalt sich mit folgenden Themen befasst:

- *Kapitel 1* beschreibt Thema, Problemstellung, Zielsetzung und Gliederung der Arbeit.

- Im *2. Kapitel* wird neben den allgemeinen Grundlagen eines Distance-Learning-Systems vor allem das System OVID in seiner Struktur und Funktionalität näher vorgestellt, um das nötige Vorwissen und Verständnis für die während dieser Arbeit entstandenen Konzeptionen zu vermitteln.

- Die Konzeptionen für die Benutzer- und Gruppenverwaltung sowie die Ressourcen-Abfrage werden im *3. Kapitel* dargelegt. Hierzu gehört neben dem weiterentwickelten Entwurf der von OVID verwendeten Datenbank, welche die Nutzerdaten verwaltet und zur Abfrage bereitstellt, auch ein Konzept zur Verwaltung und Manipulation dieser Daten über ein entsprechendes Interface. Ein weiterer Themenschwerpunkt dieses Kapitels beinhaltet die Beschreibung der Konzepte für die Interaktionskomponenten, welche im hier entworfenen Distance-Learning-System GWOTS Verwendung finden.

- Die programmiertechnische Umsetzung der im Kapitel 3 vorgestellten Konzepte ist Gegenstand der Ausführungen im *4. Kapitel*.

- Im *5. Kapitel* werden anhand von Testergebnissen Leistungsfähigkeit und Leistungsgrenzen des hier entworfenen Distance-Learning-Systems GWOTS untersucht, um Rückschlüsse für den realen Einsatz dieses Systems und die tatsächliche Umsetzung des Gesamtkonzeptes ziehen zu können. Außerdem wird das System GWOTS anhand der ermittelten Ergebnisse mit dem System OVID verglichen und bewertet.

- Eine Schlussbetrachtung dieser Arbeit sowie hieraus gewonnene Erkenntnisse bezüglich weiterführender Möglichkeiten sind dann abschließend Gegenstand des *6. Kapitels*.

Mit der hier vorliegenden Arbeit wird ein Distance-Learning-System entwickelt, welches aufgrund seiner übersichtlichen Gestaltung des Benutzerinterfaces vom Anwender leicht zu bedienen ist und dabei neben einem komfortablen Zugriff auf die digital aufbereiteten Vorlesungsinhalte auch die Zusammenarbeit von Lernenden in Gruppen via Chat, Whiteboard und Videokonferenz ermöglicht. Für alle hier integrierten Dienste werden im Rahmen dieser Arbeit Konzeptionen entworfen, die auch die Nutzung des jeweiligen Dienstes über eine schmalbandige Netzanbindung, beispielsweise eine Modemverbindung, berücksichtigen. So wird unter anderem auch die Qualität der in den Vorlesungsmaterialien integrierten Videosequenzen an die verfügbare Bandbreite der Netzanbindung eines Nutzers automatisch angepasst, um auch Anwendern mit Netzanbindungen geringer Bandbreite die Wiedergabe solcher Videosequenzen und damit überhaupt das Distance-Learning in seinem eigentlichen Umfang zu ermöglichen. Auch für die Nutzerkommunikation via Videokonferenz wird im vorliegenden System keine breitbandige Netzanbindung vorausgesetzt. Hierbei sind sowohl private Konferenzen zwischen zwei Teilnehmern als auch öffentliche Konferenzen im Rahmen einer Gruppe möglich. Außerdem werden dem Anwender mit den Zusatzdiensten für eMail, Kurznachrichten und FTP

weitere Formen der Kommunikation beziehungsweise des Datenaustausches zur Unterstützung des Distance-Learnings bereitgestellt.

Das Konzept für ein einheitliches Sitzungsmanagement sichert seinerseits eine komfortable Administrierung des Distance-Learning-Systems GWOTS. Es ermöglicht die visuelle Darstellung der angemeldeten Nutzer für den Dienst zur Präsentation der Vorlesungsinhalte sowie der Zusatzdienste Chat, Whiteboard und Videokonferenz innerhalb der grafischen Ausgabe des Server-Moduls.

Desweiteren wird mit der hier vorliegenden Arbeit eine Datenbank entworfen und implementiert, welche zur Verwaltung der Nutzerdaten dient. Die für das Server-Modul entwickelte grafische Schnittstelle ermöglicht hierfür die komfortable Einsichtnahme der darin enthaltenen Daten sowie deren Manipulation.

Kapitel 2

Grundlagen eines Distance-Learning-Systems

Die Zielstellung dieses Kapitels besteht in der Vermittlung von Grundlagenwissen im Bereich des Distance-Learning. Hierzu werden zunächst einige grundsätzliche Eigenschaften solcher Systeme herausgestellt, bevor das als Grundlage der hier vorliegenden Arbeit dienende Distance-Learning-System OVID in seiner Struktur und Funktionalität näher vorgestellt wird. Die hier aufgezählten Themenschwerpunkte dienen als Fundament für das notwendige Vorwissen und Verständnis zu den Ausführungen im *Kapitel 3*, welche im Einzelnen die im Rahmen dieser Arbeit erstellten Konzepte beschreiben.

2.1 Allgemeine Grundlagen

Wie bereits in *Kapitel 1* erwähnt, handelt es sich beim Distance-Learning um eine Weiterentwicklung traditioneller Formen des Fernunterrichts beziehungsweise Fernstudiums. Bei dieser Lernform findet das Internet als Übertragungsmedium Anwendung, welches eine Vielzahl von Möglichkeiten zur multimedialen Informationsübermittlung bereitstellt. Gerade der Informationsaustausch durch den Einsatz von Audio- und Video-Technologien ermöglicht hierbei eine Qualität der Fernlehre, wie sie beim traditionellen Fernunterricht einfach undenkbar wäre und dem realen Unterricht, beispielsweise in Form von Teilnahmen an Vorlesungen und Seminaren an einer Hochschule, nahe kommt.

Beim Distance-Learning bilden sich zwei thematische Schwerpunkte heraus, die als *Teleteaching* und *Teletutoring* bezeichnet werden [STE00]:

- **Teleteaching**: Teleteaching beinhaltet die Übertragung einer Lehrveranstaltung (z.B. eine Vorlesung oder Schulstunde) mittels Audio/Video-Technologie an mehrere Orte, welche auf der gesamten Welt verteilt sein können. Dabei handelt es sich in der Regel um eine Ein-Weg-Kommunikation, was bedeutet, dass die Daten vom Sender an die Lernen-

den übertragen werden, jedoch nicht umgekehrt. Der Informationsaustausch zwischen dem einzelnen Lernenden und dem Lehrenden erfolgt beim Teleteaching in Form von Rückmeldungen. Dies geschieht entweder asynchron, zum Beispiel per eMail, oder synchron, zum Beispiel per Videokonferenz.

- **Teletutoring**: Beim Teletutoring betreut ein Lehrender nach Bedarf einen Lernenden bei Fragen zum Lernmaterial oder Lernprozess. Der Informationsaustausch kann auch hier, genau wie beim Teleteaching, auf synchrone Weise, zum Beispiel per eMail, oder asynchrone Weise, zum Beispiel per Videokonferenz, erfolgen.

Aus der beschriebenen Problematik wird ersichtlich, dass zur Umsetzung dieser Funktionalitäten zwei verschiedene Systemkomponenten zur Anwendung kommen müssen. Eines, das die Lehrinhalte in Form digital gespeicherter Informationen zur Verfügung stellt, und ein weiteres, welches diese Informationen für den Lernenden abrufbar macht und die angesprochenen Dienste für Rückmeldungen in Form eines synchronen beziehungsweise asynchronen Informationsaustausches zwischen Lehrenden und Lernendem zur Verfügung stellt. Der Informationsaustausch zur Klärung von Fragen, welche das Lehrmaterial beziehungsweise den Lernprozess betreffen, ist auch unter den einzelnen Lernenden in Form von Gruppen möglich, so dass wie im realen Leben bei Bedarf Lerngemeinschaften unter den Lernenden gebildet werden können.

Als Basis für die Umsetzung der hier angesprochenen Funktionalitäten innerhalb eines Distance-Learning-Systems dient die Client/Server-Architektur, welche ausführlich in den Ausarbeitungen von [ROS00] und [KUR00] besprochen wird. Der darauf basierende Entwurf für das Distance-Learning-System OVID sowie die damit verbundenen Dienste zum Informationsaustausch sind Gegenstand des folgenden Abschnitts.

2.2 Das Distance-Learning-System OVID

Der auf der Client/Server-Architektur basierende Entwurf des Systems OVID umfasst zwei Komponenten, die nachfolgend kurz erläutert werden [KUR00]:

- Der **Server** (zentraler Service-Anbieter bzw. zentraler Server) hat die Aufgabe, die Benutzerverwaltung und das Sitzungsmanagement zu übernehmen. Er wird vom Systemadministrator gesteuert.

- Der **Client** (zentraler Client bzw. zentraler Dienst) stellt nach erfolgreicher Anmeldung am Server alle verfügbaren Dienste über ein grafisches Interface bereit, wobei er selbst speziell zur Wiedergabe digital gespeicherter Vorlesungsinhalte konzipiert ist. Er wird vom Anwender gestartet und gesteuert.

Der für das System OVID entworfene und im zentralen Server integrierte Sitzungsmanager kontrolliert alle Aktivitäten von Benutzern, beginnend zum Zeitpunkt ihrer Anmeldung am

System bis hin zum Verlassen desselbigen. Die Verwaltung von Benutzerdaten erfolgt hierbei mit Hilfe einer SQL-Datenbank, welche neben verschiedenen Informationen über den Nutzer auch dessen zur Anmeldung am Server erforderliches Login und Passwort beinhaltet.

Der zentrale Client des Distance-Learning-Systems OVID umfasst die folgenden Dienste:

- Wiedergabe von abgespeicherten und digital aufbereiteten Vorlesungsinhalten
- Chat-Dienst
- Whiteboard-Dienst
- Videokommunikation mit ausgewähltem Partner
- eMail
- FTP
- Kurznachrichten-Dienst.

In *Abbildung 2.1* wird die hier beschriebene Client/Server-Architektur am Beispiel des OVID grafisch dargestellt.

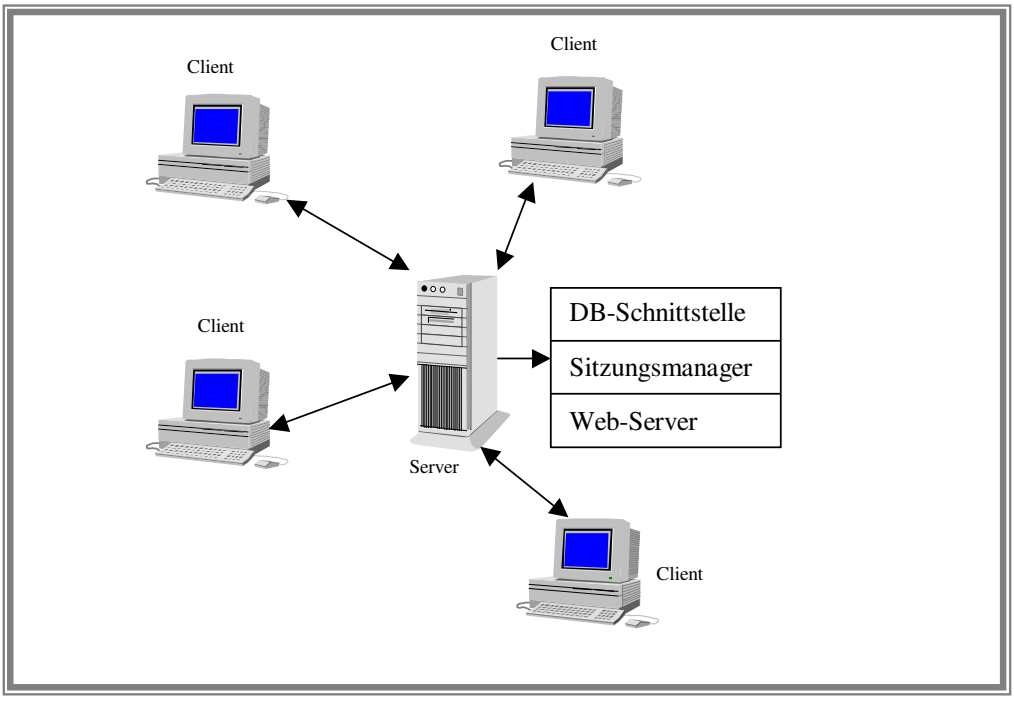

Abbildung 2.1: Schematische Darstellung des Client/Server-Modells im System OVID [KUR00]

Die Präsentation von Vorlesungsinhalten stellt beim System OVID den Hauptanwendungszweck dar. Der hierbei digital abgespeicherte Lehrstoff wird dem Anwender in Form von

HTML-Seiten angezeigt. Zu diesem Zweck kommt neben dem zentralen Server auch ein Web-Server zum Einsatz, welcher diese HTML-Seiten für den zentralen Dienst zur Verfügung stellt. Vom zentralen Server wird dem zentralen Client bei dessen erfolgreicher Anmeldung an das System die Adresse einer Startseite übermittelt, welche den eigentlichen Vorlesungsinhalt repräsentiert.

Sitzungsmanagement im OVID

Im zentralen Server ist ein Sitzungsmanager integriert, welcher sämtliche Aktivitäten, die ein Benutzer über die verschiedenen Dienste ausführen kann, kontrolliert und entsprechend Aktionen ausführt. Der zentrale Server kommuniziert im System OVID über einen TCP/IP-Socket mit dem zentralen Client, um die für das Sitzungsmanagement anfallenden Nachrichten zu empfangen beziehungsweise zu senden. Solche Nachrichten können folgenden Hintergrund besitzen [KUR00]:

- Anmeldung eines Benutzers am System und Bekanntgabe der verfügbaren Ressourcen durch den zentralen Client
- Freigabe der Dienste durch den zentralen Server
- Übermittlung der HTML-Startseite des Vorlesungsinhaltes an den zentralen Client
- Abmeldung eines Benutzers vom System.

Die *Abbildung 2.2* veranschaulicht die hierbei verwendete Struktur von zentralem Client und zentralem Server in Form von Modulen sowie deren Kommunikationsweg auf grafische Weise. Dabei wird die Struktur des Moduls des zentralen Servers in drei Gruppen gegliedert, welche neben der zur Client/Server-Kommunikation verwendeten Socket-Komponente und der zur Nutzer- und Dienstverwaltung eingesetzten Sitzungsmanagement-Komponente auch eine Schnittstelle zu einer Datenbank beschreibt. Diese Schnittstelle zur Datenbank bietet dem zentralen Server mittels der deskriptiven Standard-Datenbank-Sprache SQL Zugriff auf die in dieser Datenbank hinterlegten Nutzerdaten, welche vom zentralen Server zur Identifizierung eines Nutzers bei dessen Anmeldung an das System benötigt werden. Die Datenbank sowie der in beide Richtungen erfolgende Datenaustausch zwischen Datenbank und zentralem Server werden in der Grafik ebenfalls dargestellt. Die aus der Abbildung ersichtlichen Pfeile zwischen zentralem Client und zentralem Server beschreiben die TCP/IP-Verbindung zwischen diesen beiden Modulen durch entsprechende Socket-Komponenten, über welche der Datenaustausch zum Sitzungsmanagement in Form von Steuernachrichten zur An- beziehungsweise Abmeldung eines Nutzers erfolgt.

Die aus der *Abbildung 2.2* ersichtliche Gliederung der Struktur des Moduls des zentralen Clients beschreibt ihrerseits die folgenden Bestandteile:

- Die *Socket*-Komponente dient zum Austausch von Steuernachrichten zwischen zentralem Server und zentralem Client per TCP/IP.
- Die *Eingabeauswertung und Verwaltung* umfasst die Auswertung und Verwaltung aller Nutzereingaben, die der Anwender am zentralen Client durchführt. Hierzu gehören die Aktionen zur System-Anmeldung beziehungsweise System-Abmeldung, das Aufrufen von Zusatzdiensten sowie das Versenden von Kurznachrichten.
- Die *URL- und HTML-Ressourcen-Kontrolle* erfolgt im zentralen Client des OVID über einen integrierten Web-Browser, welcher die Lehrinhalte in optischer und akustischer Form darstellt sowie die Navigation zwischen einzelnen Abschnitten des Lehrstoffes über entsprechende Verweise innerhalb dieser HTML-Seiten realisiert.
- Die im zentralen Client vorhandenen *Routinen der einzelnen Dienste* beinhalten die Operationen, welche zum Starten der hierzu entsprechend vorhandenen Dienst-Client-Module erforderlich sind. Diese Module stellen die Dienste Chat, FTP, eMail, Whiteboard, Video-Konferenz und Nachrichtenaustausch in Form von Kurznachrichten bereit.

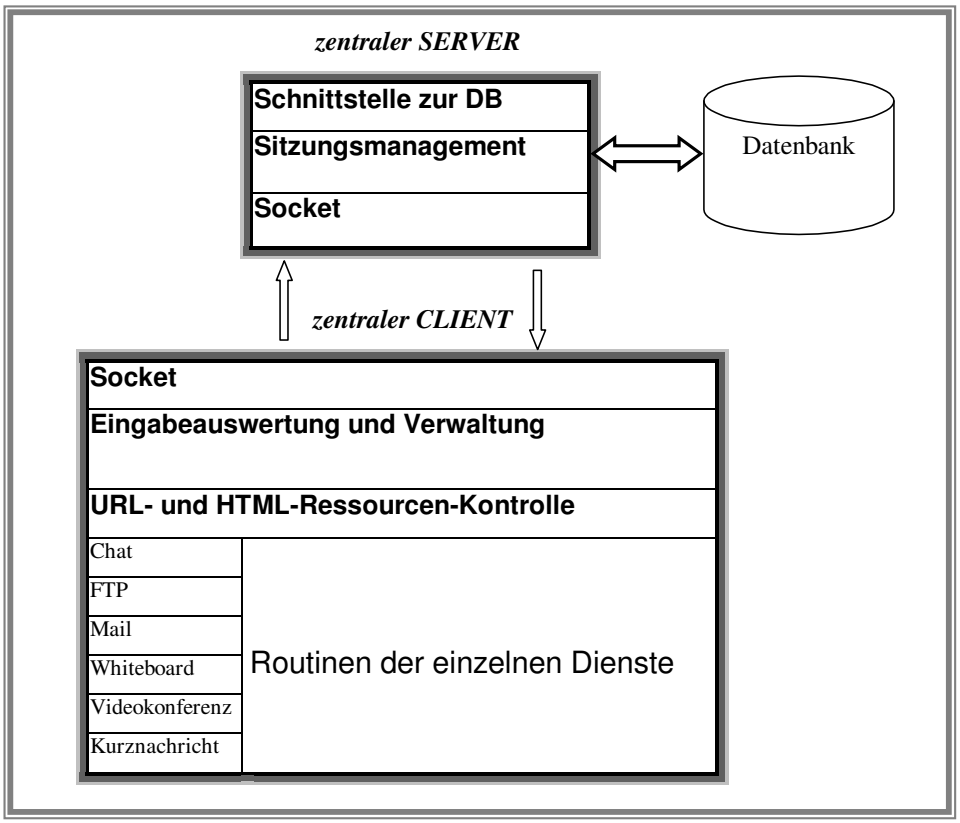

Abbildung 2.2: Schematische Darstellung der Architektur im OVID [KUR00]

Alle neben dem Dienst zur Wiedergabe der Vorlesungsinhalte im System OVID verfügbaren Interaktionskomponenten stellen Zusatzdienste dar, welche vom Benutzer zur Unterstützung des Lernprozesses verwendet werden können und hierzu aus der Oberfläche des zentralen Client heraus gestartet werden. Ein Teil dieser Zusatzdienste wird in Form von Modulen eingebunden, welche unabhängig vom System OVID als eigenständig lauffähige Komponenten verwendbar sind. Hieraus ergibt sich der Umstand, dass beispielsweise das eigenständig lauffähige Modul für den Chat-Dienst entsprechend der Client/Server-Architektur ebenfalls aus zwei Komponenten besteht, nämlich einem eigenen Service (Chat-Server) und einem Client (Chat-Client). Dieser Chat-Server arbeitet hierbei unabhängig vom zentralen Server.

Eine zentrale Zielstellung der Konzeption des Systems OVID ist ein einheitliches Sitzungsmanagement, welches gegenüber einem verteilten Sitzungsmanagement den Vorteil mit sich bringt, dass alle Aktivitäten der Benutzer, unabhängig vom verwendeten Dienst, auf einen Blick nachvollziehbar sind und vom zentralen Server gesteuert werden können. Um dieses einheitliche Sitzungsmanagement auch bei Verwendung der zuvor beschriebenen eigenständig lauffähigen Module realisieren zu können, ist der zentrale Server im System OVID in der Lage, Steuernachrichten an die entsprechenden Dienst-Client-Module, wie beispielsweise den Chat-Client, zu senden und ebensolche von diesen zu empfangen. Hierzu baut das Dienst-Client-Modul, so bald es aktiv ist, einen Kommunikationkanal zum zentralen Server auf und sendet und verarbeitet seinerseits Steuernachrichten. [KUR00].

Folgende Arten von Steuernachrichten können im System OVID auftreten:

- Beitritt in eine Gruppe
- Verlassen einer Gruppe
- Aufforderung zum Verlassen eines Dienstes durch den zentralen Server.

Im System OVID handelt es sich bei den angesprochenen Gruppen um sogenannte Chat-Gruppen beziehungsweise Whiteboard-Gruppen. Diese setzen sich aus einem oder mehreren an das System angemeldeten Nutzern zusammen, welche innerhalb der betreffenden Gruppe eine gemeinsame Diskussion per Chat-Dienst führen beziehungsweise an einem gemeinsamen Projekt mittels des integrierten Whiteboard arbeiten. In [KUR00] werden detailierte Ausführungen zur Gruppenkommunikation gemacht, die an dieser Stelle jedoch nicht weiter betrachtet werden.

Mit Hilfe der *Abbildung 2.3* wird die soeben besprochene Problematik der Kommunikationsvorgänge beim Zusammenspiel zwischen Dienst-Client und zentralem Server auf grafische Art veranschaulicht. Die dargestellten Module des zentralen Servers und zentralen Clients sowie deren Kommunikationsweg entsprechen den Ausführungen zu *Abbildung 2.2*, wobei das Modul des zentralen Client in *Abbildung 2.3* allerdings etwas vereinfacht dargestellt wird. Die internen Routinen beinhalten dabei die in *Abbidlung 2.2* dargestellten und hierzu

beschriebenen Bestandteile zur Eingabeauswertung und Verwaltung, zur URL- und HTML-Ressourcenkontrolle sowie zum Aufruf der einzelnen Zusatzdienste für Chat, FTP, eMail, Whiteboard, Videokonferenz und Kurznachricht.

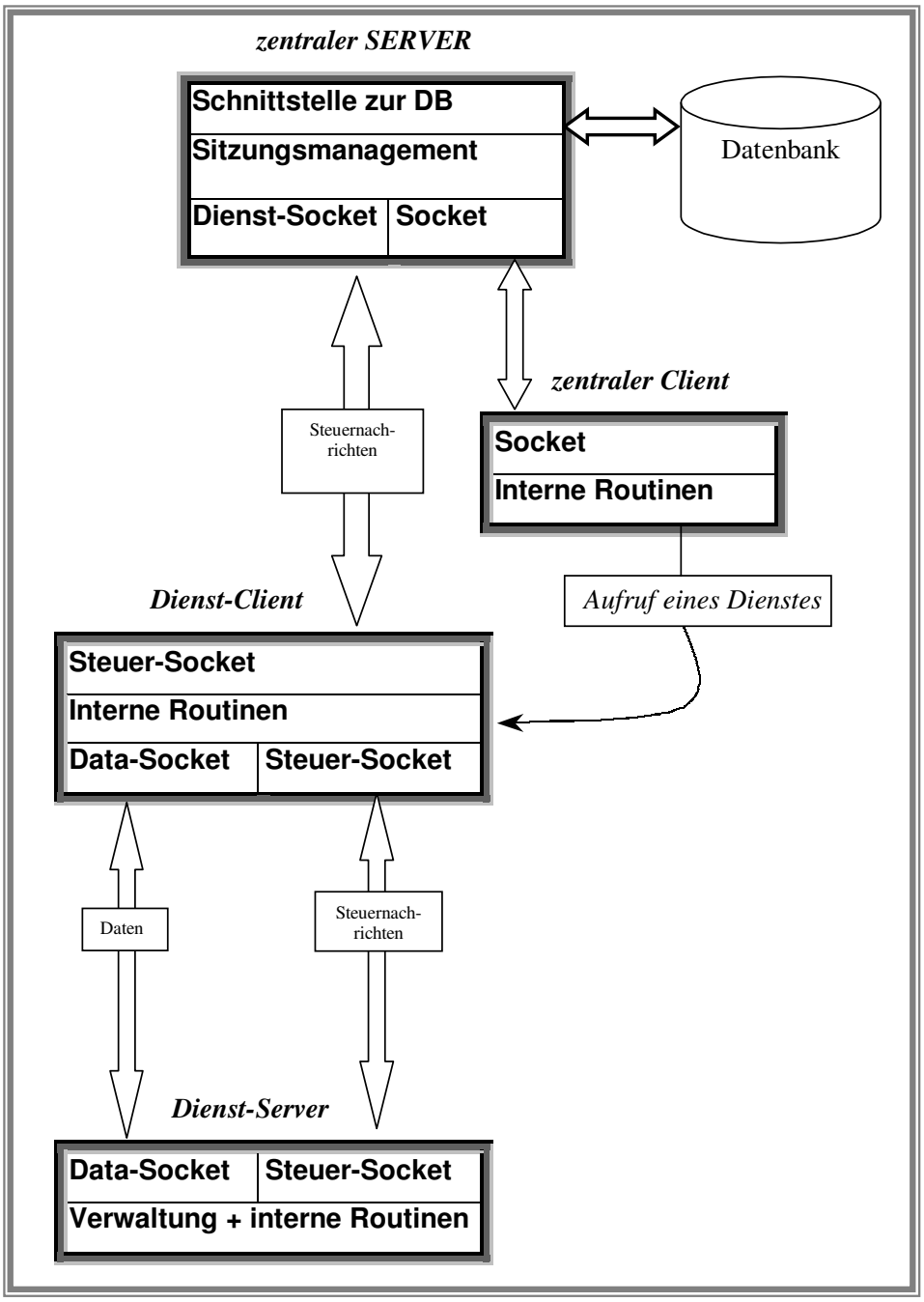

Abbildung 2.3: Strukturierung der Systemmodule und Zusammenspiel von Dienst-Client, Dienst-Server, zentralem Client und zentralem Server [KUR00]

Der in *Abbildung 2.3* dargestellte Ablauf zwischen zentralem Client und dem Modul des Dienst-Clients verhält sich dabei so, dass ein Nutzer nach erfolgreicher Anmeldung an den zentralen Server über den zentralen Client die verfügbaren Dienst-Clients der Zusatzdienste aufrufen kann. Dabei wird der Auswahl entsprechend auf dem Anwender-Rechner der zugehörige Dienst-Client gestartet, welcher sich daraufhin sowohl mit dem korrespondierenden Dienst-Server als auch mit dem zentralen Server über die hierfür vorhandenen Sockets in Verbindung setzt. Mit dem Dienst-Server tauscht er dann solange Daten und Steuernachrichten über den hierfür vorhandenen Data- beziehungsweise Steuer-Socket aus, bis der Anwender die Nutzung des Zusatzdienstes beendet. Der Austausch von Steuernachrichten zwischen zentralem Server und Dienst-Client erfolgt über den hierfür vorhandenen Steuer-Socket des Dienst-Clients sowie den Dienst-Socket des zentralen Servers und beschränkt sich dabei auf die Informationen über die An- beziehungsweise Abmeldung an den entsprechenden Dienst, welche dem Sitzungsmanager des zentralen Servers zugeführt und von diesem ausgewertet werden. Während die internen Routinen des Dienst-Clients beziehungsweise Dienst-Servers die dem entsprechenden Zusatzdienst zugehörigen Funktionalitäten beinhalten, beschreibt der Bestandteil der Verwaltung des Dienst-Servers den darin enthaltenen Sitzungsmanager, welcher die Gruppen und Nutzer des jeweiligen Dienstes verwaltet. Der in *Abbildung 2.3* dargestellte Dienst-Client beziehungsweise Dienst-Server steht stellvertretend für eines der Dienst-Client-Module beziehungsweise Dienst-Server Module der im OVID verfügbaren Zusatzdienste, welche auf der Client/Server-Architektur basieren.

Anhand der in *Abbildung 2.3* dargestellten Kommunikationsvorgänge werden folgende Zusammenhänge sichtbar:

- Eigenständigkeit von Dienst-Client und Dienst-Server gegenüber zentralem Server
- Aufruf des Dienst-Clients aus dem zentralen Client heraus
- Austausch von Steuernachrichten und Daten zwischen Dienst-Client und Dienst-Server zum Zweck der Umsetzung der Funktionalitäten des jeweiligen Dienstes
- Austausch von Steuernachrichten zwischen Dienst-Cient und zentralem Server zur Umsetzung eines einheitlichen Sitzungsmanagements.

Datenverwaltung im OVID

Neben der Architektur wird im Rahmen dieses Kapitels noch die Art und Weise der Verwaltung von Nutzerdaten im System OVID in den Grundzügen erläutert, da dies ein untrennbarer Bestandteil des Sitzungsmanagements ist. Hierzu verwaltet der im OVID integrierte Sitzungsmanager eine SQL-fähige Datenbank. Deren Struktur wird zunächst vorgestellt, bevor einige Aussagen zu den wichtigsten Datenfeldern gemacht werden. Die Datenbank beinhaltet alle in der *Tabelle 2.1* aufgeführten Datenfelder, welche in einer einzigen Relation definiert sind.

Die Relation *Profile* mit ihren Attributen			
Feld	**Beschreibung**	**Feldtyp**	**Info**
Login	Kennung	VarChar(10)	Primärschlüssel
Pass	Passwort	VarChar(10)	
Name	Nachname	VarChar(30)	
Vorname	Vorname	VarChar(30)	
Fach	Fachbereich	VarChar(50)	
Matr	Matrikel	VarChar(10)	
Mail	eMail-Adresse	VarChar(50)	

Tabelle 2.1: Struktur der Datenbanktabelle zur Benutzerverwaltung im System OVID

Wie hieraus ersichtlich ist, dient die Nutzerkennung als Primärschlüssel und ermöglicht somit eine eindeutige Identifizierung von Benutzern innerhalb des Systems. Aus Gründen der Sicherheit und der Wahrung der Identität ist zur erfolgreichen Anmeldung eines Nutzers an das Distance-Learning-System außerdem neben der Nutzerkennung noch die Angabe eines zugehörigen Passwortes erforderlich, welches zur Gegenprüfung vom System ebenfalls in der Datenbank hinterlegt ist. Alle weiteren in *Tabelle 2.1* aufgeführten Daten werden vom Sitzungsmanager nicht ausgewertet und dienen lediglich dazu, dem Administrator des Systems die personenbezogenen Informationen über einen Nutzers zur Ansicht zu bringen, falls er diese anfordert.

Im Rahmen des Sitzungsmanagements spielt die Nutzerkennung eine tragende Rolle. Sie dient nämlich nicht nur zur Identifizierung eines Nutzers bei dessen Anmeldung an das System, sondern wird vom Sitzungsmanager auch zur Zuordnung des zugehörigen Nutzers zu Interaktionsdiensten wie dem Chat-Dienst oder dem Whiteboard-Dienst verwendet.

Kapitel 3

Konzeption des Distance-Learning-Systems GWOTS

Aufbauend auf den Ausführungen des vorangegangenen Kapitels werden im Folgenden die während dieser Arbeit entwickelten Konzeptionen vorgestellt. Das Ziel des Gesamtkonzeptes dieser Arbeit stellt dabei in Hinblick auf den Funktionsumfang keineswegs die Erweiterung des Systems OVID dar, sondern vielmehr steht die Entwicklung und Erprobung eines alternativen Distance-Learning-Systems im Vordergrund, welches die Nutzung aller vom GWOTS angebotenen Dienste auch bei schmalbandigen Netzanbindungen ermöglicht und somit sowohl die Praxistauglichkeit dieses Systems als auch die Anwenderakzeptanz erhöht.

Das Distance-Learning-System GWOTS besitzt die folgenden Eigenschaften, welche in den weiteren Ausführungen noch genauer beschrieben werden:

- Sowohl die Möglichkeiten der Netzwerke mit einer hohen Bandbreite als auch der Einsatz des Systems in den schmalbandigen Netzwerken werden für alle angebotenen Dienste unterstützt.
- Alle angemeldeten Benutzer können das System in Anspruch nehmen. Der Standort des Benutzers spielt dabei keine Rolle, lediglich ein Internet-Zugang ist erforderlich. Spezielle Hardware wird nicht benötigt.
- Das System stellt zur Unterstützung des Lernprozesses Werkzeuge zur Gruppenarbeit in Form der Dienste Chat, Whiteboard und Videokonferenz zur Verfügung.
- Das System stellt desweiteren die Dienste FTP, eMail und Kurznachricht zum Datenbeziehungsweise Informationsaustausch bereit.
- Das System besitzt eine modulare Struktur, um die Weiterentwicklung einfach zu gestatten.

- Das System entspricht den ergonomischen Anforderungen und ist leicht zu warten, um die nötige Akzeptanz beim Nutzer zu finden.

Aufgrund der gestellten Anforderungen und Eigenschaften wird das System GWOTS auf Grundlage der Client/Server-Architektur entworfen, welche dem Anwender über hierfür vorhandene Client-Komponenten die Nutzung der verschiedenen Dienste des Systems gestattet. Zur Bereitstellung und Verwaltung dieser Dienste sind korrespondierend zu den Client-Komponenten entsprechende Server-Komponenten vorhanden. Diese im Distance-Learning-Systems GWOTS verfügbaren Client-seitigen und Server-seitigen Komponenten sind Gegenstand des in *Abbildung 3.1* dargestellten Dienstemodells.

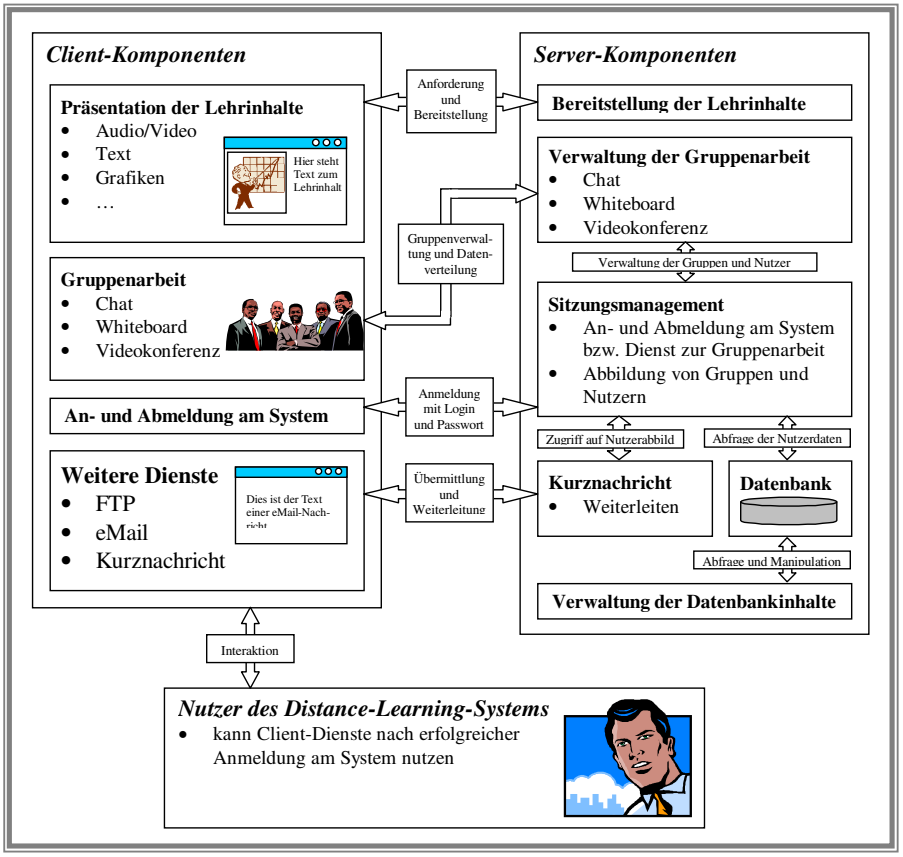

Abbildung 3.1: Dienstemodell entsprechend der Client/Server-Architekur im GWOTS

Aus der Darstellung in *Abbildung 3.1* geht hervor, dass die Client-Komponenten auch eine Komponente zur An- und Abmeldung des Nutzers am System beinhalten. Diese übermittelt bei der Anmeldung an das System zur Identifikation des Nutzers ein vom diesem angegebenes Login und Passwort an die Server-seitige Sitzungsmanagement-Komponente und erhält daraufhin von dieser ein positives oder negatives Ergebnis bezüglich der Verifikation dieser Daten. Bei erfolgreicher Anmeldung werden die Client-seitigen Dienste zur Verwendung freigeschaltet und der Nutzer mittels der Sitzungsmanagement-Komponente auf das System abgebildet,

so dass seine Aktivitäten bei der Nutzung dieser Diensten im weiteren Verlauf verfolgt und verwaltet werden können. Die Verifikation der bei der Anmeldung übermittelten Daten erfolgt anhand eines Vergleiches mit den hierzu korrespondierenden Daten des entsprechenden Nutzers, welche in einer Datenbank hinterlegt sind, auf die die Sitzungsmanagement-Komponente über eine hierfür vorhandene Schnittstelle zugreift.

Für die Nutzung des Dienstes zur Präsentation der Lehrinhalte ist Server-seitig eine Komponente zur Bereitstellung der Lehrinhalte vorhanden, welche der Client-Komponente dieses Dienstes auf deren Anforderung die gewünschten Daten übermittelt, woraufhin diese dem Nutzer Client-seitig zur Ansicht gebracht werden. Die Auswahl der angeforderten Lehrinhalte übernimmt dabei der Nutzer, indem er mit der Client-Komponente dieses Dienstes interagiert.

Die Werkzeuge zur Gruppenarbeit, welche die Dienste Chat, Whiteboard und Videokonferenz beinhalten, werden ebenfalls durch Interaktion mit dem Nutzer gesteuert. Dabei werden alle Input-Informationen eines Nutzers des jeweiligen Dienstes an die korrespondierende Server-Komponente übermittelt und von dieser gruppenbezogen an die entsprechende Client-Komponente anderer Nutzer, welche ebenfalls diesen Dienst nutzen, weitergeleitet und dort als Output-Information ausgegeben. Gruppenbezogen bedeutet, dass die Dienste zur Gruppenarbeit jeweils mehrere Gruppen verwalten können und die Input-Informationen eines Nutzers, der genau einer Gruppe angehört, nur an die Nutzer dieses Dienstes weitergeleitet werden, die ebenfalls zu dieser Gruppe gehören. Zur Verwaltung von Nutzern und Gruppen nutzen die Dienste zur Gruppenarbeit deshalb ebenfalls die Sitzungsmanagement-Komponente, welche die Nutzer und Gruppen des jeweiligen Dienstes auf das System abbildet.

Der Dienst zum Austausch von Kurznachrichten stellt Server-seitig eine Komponente bereit, die solche von einem Nutzer mit Hilfe der korrespondierenden Clinet-Komponente verfassten Kurznachrichten empfängt und an die entsprechende Client-Komponente des adressierten Nutzers weiterleitet. Dabei wird auf die Sitzungsmanagement-Komponente zugegriffen, um anhand des dort abgebildeten Nutzers, für den die Kurznachricht bestimmt ist, dessen zur Übermittlung benötigte Adresse zu ermitteln.

Zur Nutzung der Client-seitig vorhanden Komponenten der Dienste FTP und eMail sind ebenfalls Server-Komponenten erforderlich, welche die Anforderungen dieser Client-Komponenten bearbeiten. Diese Server-Komponenten sind allerdings nicht fester Bestandteil des Systems GWOTS, sondern werden durch beliebige im Internet verfügbare FTP- beziehungsweise Mail-Server dargestellt. Die Client-seitige Bereitstellung hat den Hintergrund, dem Anwender aus dem Distance-Learning-System heraus den komfortablen Zugriff auf solche externen Server-Komponenten zu ermöglichen.

Zur Verwaltung der in der Datenbank enthaltenen Nutzerdaten, welche sowohl die Ansicht als auch die Manipulation dieser Daten beinhaltet, steht Server-seitig die Komponente zur Verwaltung der Datenbankinhalte zur Verfügung. Diese besitzt hierzu eine Schnittstelle zur Datenbank.

Alle im GWOTS angebotenen Dienste können vom Anwender parallel genutzt werden. Für die Dienste zur Gruppenarbeit gilt hierbei, dass ein Anwender innerhalb des jeweiligen Dienstes bei dessen Nutzung gleichzeitig immer nur in einer einzigen Gruppe aktiv sein kann. Vor dem Wechsel in eine andere Gruppe des gleichen Dienstes muss er deshalb zunächst die aktuell zugehörige Gruppe verlassen, bevor er der anderen Gruppe beitreten kann.

Die Kommunikationswege und -formen zwischen den einzelnen Komponenten des Systems werden in *Abbildung 3.2* beschrieben. Hieraus geht hervor, dass zur Kommunikation beziehungsweise zum Datenaustausch zwischen den Server-Komponenten und den Client-Komponenten TCP/IP-Verbindungen verwendet werden. Die Server-Komponenten für die Dienste Chat, Whiteboard, Videokonferenz und Kurznachricht tauschen Daten mit der Sitzungsmanagement-Komponente aus, um ihre Nutzer beziehungsweise Nutzergruppen auf das System abbilden und somit verwalten zu können. Die Client-Komponente zur An- und Abmeldung eines Nutzers am System tauscht hierzu direkt per TCP/IP-Verbindung die betreffenden Daten mit der Sitzungsmanagement-Komponente aus. Außerdem wird in *Abbildung 3.2* noch die Interaktion zwischen dem Nutzer des Distance-Learning-Systems und den Client-Komponenten dargestellt, welche die Client-seitige Ein- und Ausgabe von Informationen durch den beziehungsweise an den Nutzer beinhaltet.

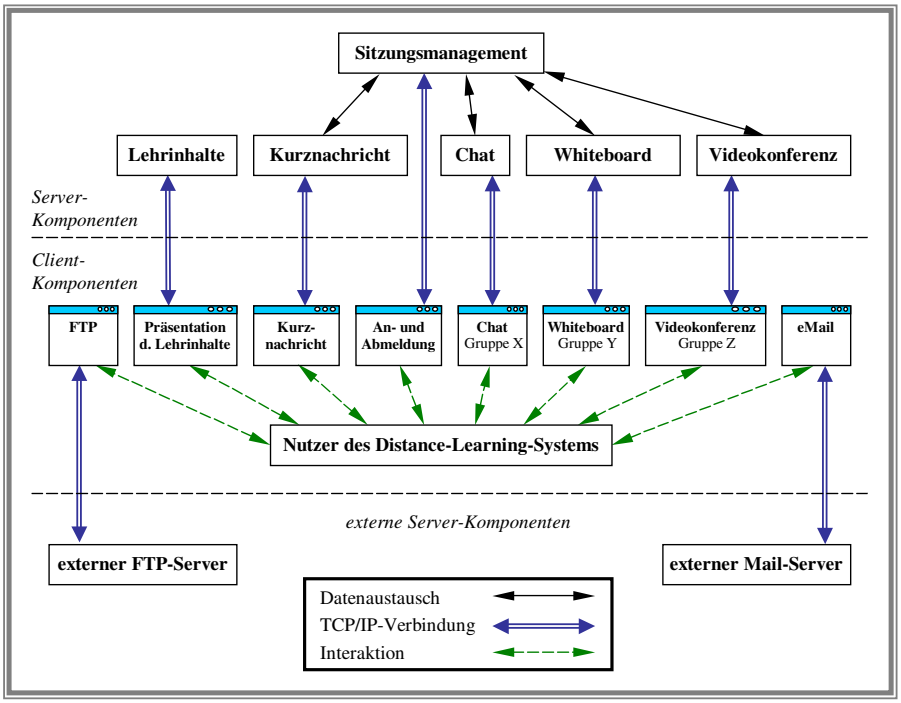

Abbildung 3.2: Darstellung der Kommunikationswege und -formen zwischen den einzelnen Komponenten des Systems GWOTS

3.1 Benutzer-, Gruppen- und Dienstverwaltung

Das Konzept zum Sitzungsmanagement beinhaltet die Verwaltung von Nutzern und Gruppen, welche an das System beziehungsweise einen der Zusatzdienste angemeldet sind. Der hierzu konzipierte Sitzungsmanager stellt alle Funktionalitäten bereit, welche eine datentechnische Abbildung dieser Nutzer und Gruppen auf das System realisieren und diese Informationen dem zentralen Server beziehungsweise den Dienst-Server-Modulen auf Abfrage bereitstellen.

Im Rahmen dieses Projektes wird zudem ein Konzept zur Strukturierung der Datenbank vorgestellt, welche die zur Verwaltung erforderlichen Nutzerdaten enthält. In unmittelbarem Zusammenhang hierzu wird außerdem ein Konzept zur Verwaltung und Manipulation der in der Datenbank gespeicherten Nutzerdaten über ein entsprechendes Interface vorgestellt, welches eine komfortable Nutzerdatenverwaltung für den Administrator möglich macht. Bevor diese Konzeptionen für die Datenverwaltung jedoch näher vorgestellt werden, folgt zunächst das Konzept zum Sitzungsmanagement.

3.1.1 Sitzungsmanagement

Im Distance-Learning-System GWOTS sind zur Unterstützung des Lernprozesses neben dem Präsentationsdienst, welcher zu seiner Verwendung eine Anmeldung des Nutzers mit Login und Passwort voraussetzt und verbunden hiermit verwaltungstechnische Aufgaben nach sich zieht, auch Zusatzdienste zur Nutzerkommunikation und zum Datentransfer integriert. Diese beinhalten die Funktionalitäten der Dienste für Chat, Whiteboard, Videokonferenz, FTP, eMail, und Kurznachrichten, wobei die Dienste für die Nutzerkommunikation via Chat, Whiteboard und Videokonferenz Werkzeuge darstellen, welche eine synchrone Zusammenarbeit zwischen Nutzern im Rahmen von Arbeitsgruppen ermöglichen und dabei ebenfalls eine datentechnische Verwaltung dieser Gruppen und Nutzer im Rahmen des Sitzungsmanagements bedingen. Im GWOTS erledigt diese Aufgabe der Sitzungsmanager des zentralen Servers, wobei die gemeinsame Nutzung dieses Sitzungsmanagers durch den zentralen Server und die Dienst-Server-Module der angesprochenen Zusatzdienste ein einheitliches Sitzungsmanagement gewährleistet, welches keine zusätzlichen Kommunikationskanäle zwischen den Dienst-Client-Modulen und dem zentralen Server erfordert.

Voraussetzung für die gemeinsame Nutzung des im zentralen Server integrierten Sitzungsmanagers durch die angesprochenen Zusatzdienste ist die Integration der Dienst-Server-Module im zentralen Server, wobei das Dienst-Server-Modul zur Videokommunikation hierbei eine Ausnahme bildet. Dieser besitzt seinen eigenen Sitzungsmanager und wird im Gegensatz zu den Dienst-Server-Modulen für Chat und Whiteboard nicht in den zentralen Server integriert, da er aufgrund der bei einer Videokommunikation erforderlichen Bandbreite bei entsprechender Frequentierung auch auf einem vom zentralen Server getrennt laufenden Rechnersystem einsetzbar sein muss, um so die Netzanbindung des zentralen Servers zu entlasten (vgl. *Abbil-*

dung 3.3).

Eine zweite Voraussetzung besteht in der Integration der Dienst-Client-Module in den zentralen Client, um zusätzliche Kommunikationskanäle zum Austausch von Steuernachrichten zwischen den Dienst-Client-Modulen und dem zentralen Server zur Realisierung eines einheitlichen Sitzungsmanagements zu vermeiden (vgl. *Abbildung 2.3* und *Abbildung 3.3*). Diese Integration erlaubt nebenbei auch die vollständige Einbindung der grafischen Ausgabeinformationen der hier aufgezählten Dienste in das Benutzerinterface des zentralen Dienstes. Der Vorteil dieses Konzeptes liegt somit neben einer einfachen Kommunikationsstruktur auch in einem kompakten Erscheinungsbild des zentralen Clients, welcher sich besonders dann positiv bemerkbar macht, wenn ein Nutzer neben dem Distance-Learning-System noch eine ganze Reihe weiterer Anwendungen geöffnet hat und dabei des öfteren zwischen den Diensten des Distance-Learning-Systems und diesen anderen Anwendungen wechselt.

Die *Abbildung 3.3* zeigt die Strukturierung der Systemmodule entsprechend der zugrundeliegenden Client/Server-Architektur sowie deren Zusammenspiel im Rahmen des Sitzungsmanagements und Datenaustausches. Das Modul des zentralen Servers gliedert sich in die folgenden Bestandteile:

- Die *Schnittstelle zur Datenbank* dient dem Zugriff auf die in der abgebildeten Datenbank hinterlegten Nutzerdaten durch den zentralen Server mittels der deskriptiven Standard-Datenbank-Sprache SQL. Diese Nutzerdaten werden vom zentralen Server zur Identifizierung eines Nutzers bei dessen Anmeldung an das System benötigt.

- Die Komponente zum *Sitzungsmanagement* dient der Abbildung und Verwaltung der am System angemeldeten Nutzer und Gruppen.

- Die *Dienst-Server-Module* beinhalten die Server-seitigen Funktionalitäten für die im zentralen Server integrierten Zusatzdienste Chat und Whiteboard. Diese verwenden zur Verwaltung der an diese Dienste angemeldeten Nutzer und Gruppen den Sitzungsmanager des zentralen Servers.

- Die *Socket*-Komponente dient dem Austausch von Steuernachrichten zwischen dem zentralen Client und dem zentralen Server, welche vom Sitzungsmanager des zentralen Servers verarbeitet werden.

- Über die *Dienst-Sockets* des zentralen Servers erfolgt der Datenaustausch zwischen den Dienst-Client-Modulen und den korrespondierenden Dienst-Server-Modulen für die Zusatzdienste Chat und Whiteboard.

Das Modul des zentralen Clients gliedert sich ebenfalls in mehrere Bestandteile mit folgendem Inhalt:

- Die *Socket*-Komponente und die *Dienst-Sockets* dienen dem bereits beschriebenen Austausch von Steuernachrichten und Daten.
- Die *Dienst-Client-Module* kapseln die Client-seitigen Funktionalitäten für die im zentralen Client integrierten Zusatzdienste für Chat, Whiteboard und Videokonferenz.
- Der *Videokonferenz-Dienst-Socket* dient dem Austausch von Daten und Steuernachrichten zwischen dem im zentralen Client integrierten Dienst-Client-Modul dieses Dienstes und dem vom zentralen Server unabhängig lauffähigen Videokonferenz-Server. Dabei informiert das Dienst-Client-Modul des Videokonferenz-Dienstes über die Socket-Komponente des zentralen Clients die im zentralen Server enthaltene Sitzungsmanagement-Komponente bezüglich der Nutzung dieses Dienstes, um auch für den Videokonferenz-Dienst trotz seiner Unabhängigkeit vom zentralen Server ein einheitliches Sitzungsmanagement zu gewährleisten.

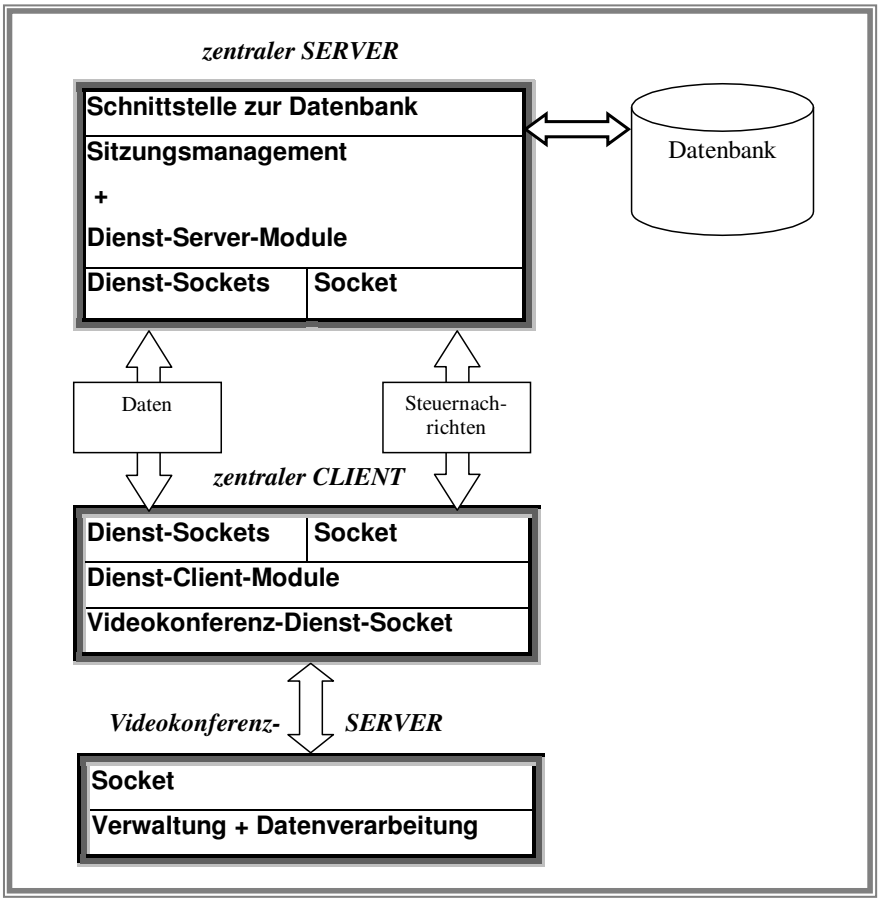

Abbildung 3.3: Zusammenspiel der Systemmodule des zentralen Clients, zentralen Servers und Videokonferenz-Servers im Rahmen des Sitzungsmanagements

Das in *Abbildung 3.3* dargestellte und vom zentralen Server unabhängig lauffähige Modul des Videokonferenz-Servers besitzt seinerseits Komponenten zur Kommunikation via Sockets, zur eigenständigen Verwaltung von Nutzern und Gruppen sowie zur Datenverarbeitung, wobei die Datenverarbeitung die Verteilung von eingehenden Audio/Video-Informationen an die Nutzer der zugehörigen Gruppe beinhaltet.

Das Prinzip der modularen Struktur des Systems beruht auf der Kapselung der im Rahmen dieser Arbeit entworfenen Interaktionsdienste und ihrer Funktionalitäten in entsprechenden Modulklassen, welche über eine fest definierte Schnittstelle genutzt und zur Zeit der Erzeugung des Programm-Codes in das Distance-Learning-System eingebunden werden. Eine Einbindung der Module zur Laufzeit ist ebenfalls realisierbar, wenn die Dienste in Form von ActiveX-Steuerelementen implementiert werden. Durch den hier beschriebenen modularen Aufbau werden im vorliegenden Konzept die Wiederverwertbarkeit und die Möglichkeit zur einfachen Weiterentwicklung sichergestellt.

Anmeldung an das System

Bei einer Anmeldung an den zentralen Server des GWOTS wird zunächst eine TCP/IP-Verbindung zwischen dem zentralen Client und dem zentralen Server aufgebaut, welche bei erfolgreicher Anmeldung auch bis zum Sitzungsende, also dem Verlassen des Systems, aufrechterhalten wird. Über diese Verbindung werden als erstes das vom Anwender angegebene Login, Passwort sowie Angaben über die verfügbaren Ressourcen an den zentralen Server übermittelt, um im nächsten Schritt von diesem verifiziert zu werden. Die Verifikation von Login und Passwort erfolgt dabei anhand der Daten, die für den entsprechenden Nutzer in der Datenbank hinterlegt sind. Sollten die übermittelten Anmeldedaten nicht korrekt sein, so wird die angeforderte Verbindung zurückgewiesen. Bei erfogreicher Verifikation hingegen wird der Benutzer in das System aufgenommen, wobei vom zentralen Server folgende Daten an den Client übermittelt werden:

- Liste der Dienste, die auf dem zentralen Client verfügbar sind
- URL, welche die Startseite für den Vorlesungsstoff beinhaltet
- Liste aller am System angemeldeten Nutzer, um Kurznachrichten an diese verschicken beziehungsweise eine Videokonferenz mit ihnen aufbauen zu können
- Liste aller Chat-Gruppen und der dazugehörigen Nutzer
- Liste aller Whiteboard-Gruppen und der dazugehörigen Nutzer
- Liste aller Videokonferenz-Gruppen und der dazugehörigen Nutzer
- IP-Adresse des Videokonferenz-Servers, da dieser auch auf einem anderen Rechner laufen kann, als der zentrale Server.

Die Liste der verfügbaren Dienste wird vom zentralen Client nach dem Erhalt ausgewertet und entsprechend umgesetzt. Welche Dienste in dieser Liste als verfügbar gekennzeichnet sind, entscheidet der zentrale Server anhand der anfangs vom zentralen Client übermittelten Angaben zu dessen Ressourcen.

Die vom zentralen Server übermittelte Startseite wird sofort an den im zentralen Client integrierten Web-Browser übergeben, um dem Anwender unmittelbar nach erfolgreicher Anmeldung den Vorlesungsinhalt zu präsentieren. Der Inhalt dieser Startseite ist hierbei vom Matrikel abhängig, welchem der Nutzer angehört und was auch aus den Nutzerdaten hervorgeht. Hierzu wird in der vom zentralen Server verwendeten Datenbank zu jedem Matrikel die entsprechende Startseite hinterlegt.

Um dem Anwender das Versenden von Kurznachrichten oder die Kommunikation via privater Videokonferenz zu ermöglichen, wird diesem eine Liste mit allen hierfür zur Verfügung stehenden Kommunikationpartnern bereitgestellt. Diese wird nach erstmaliger Übermittlung an den zentralen Client während der gesamten Sitzungsdauer vom zentralen Server über entsprechende Steuernachrichten aktualisiert, sobald ein Nutzer dem System beitritt oder es verlässt, so dass der Anwender immer auf dem aktuellsten Stand ist.

Für die Zusatzdienste Chat, Whiteboard und Videokonferenz werden dem Anwender entsprechend Listen mit den verfügbaren Chat-Gruppen, Whiteboard-Gruppen und Konferenz-Gruppen übermittelt. Um dem Anwender auch die Möglichkeit zu geben, sich einen Überblick darüber zu verschaffen, wer in welcher Gruppe aktiv ist, erfolgt für diese Dienste außerdem noch die Übertragung der zu den jeweiligen Gruppen gehörigen Nutzerkennungen in Listenform. Die nach erstmaliger Übermittlung erforderliche Aktualsierung der hier beschriebenen Listen erfolgt analog zur Aktualiserung der Liste für Kurznachrichten und private Videokonferenzen, nur dass hier der Beitritt beziehungsweise das Verlassen eines Nutzers in beziehungsweise aus einer Gruppe der Auslöser für das Versenden einer entsprechenden Steuernachricht seitens des zentralen Servers ist.

Gruppenverwaltung

Der Sitzungsmanager verwaltet für den zentralen Server eine Liste mit Nutzerobjekten, die sämtliche an das System angemeldete Nutzer abbilden. Ein solches Nutzerobjekt beinhaltet dabei die nachfolgend auszugsweise aufgelisteten Eigenschaften, welche dem Sitzungsmanager anhand der darin enthaltenen Informationen unter anderem die Zuordnung eines Nutzers zu einer Gruppe und zum Socket des jeweiligen Dienstes erlaubt:

- Login
- zugehörige Chat-Gruppe
- zugehörige Whiteboard-Gruppe

- zugehörige Videokonferenz-Gruppe
- zugehöriger Socket des zentralen Servers
- zugehöriger Chat-Socket
- zugehöriger Whiteboard-Socket
- ...

Außerdem verwaltet der Sitzungsmanager für die Zusatzdienste Chat, Whiteboard und Videokonferenz auch jeweils eine Liste von Gruppenobjekten, wobei jedes Gruppenobjekt wiederum eine Liste von Nutzerobjekten verwaltet. Somit kann der Sitzungsmanager entsprechend des gewählten Dienstes auf alle darin enthaltenen Gruppen und deren zugehörige Nutzer zugreifen. Dabei gehört ein Nutzer zu keiner, einer oder mehreren Gruppen. Ebenso können zu einer Gruppe kein, ein oder mehrere Nutzer gehören. Eine Gruppe gehört genau zu einem Dienst, wobei einem Dienst wiederum keine, eine oder mehrere Gruppen angehören können. Jedes Gruppenobjekt enthält folgende Informationen:

- Name der Gruppe
- Dienstart (z.B. Chat)
- Uhrzeit der Eröffnung der Gruppe
- Liste der Teilnehmer
- Liste der Grafiken (nur bei Whiteboard).

Um einem Nutzer beim Beitritt in eine Whiteboard-Gruppe den aktuellen Stand deren Arbeit zur Verfügung stellen zu können, wird das Gruppenobjekt im vorliegenden Entwurf um die angegebene Liste erweitert. Damit wird dem Konzept des hier entwickelten Whiteboard-Dienstes *(Kapitel 3.3.4)* Rechnung getragen.

Verlassen eines Dienstes oder des Systems

Im GWOTS gibt es zwei Möglichkeiten zum Verlassen des Systems, welche nachfolgend aufgelistet werden:

- Der Benutzer beendet den Dienst.
- Der Administrator entfernt einen Benutzer aus einem Dienst.

Beim Verlassen eines Dienstes durch den Nutzer wird vom zentralen Client eine entsprechende Steuernachricht an den zentralen Server geschickt, damit der darin integrierte Sitzungsmanager über diesen Vorgang informiert wird. Daraufhin informiert der zentrale Server alle am

System angemeldeten Clients über diesen Abmeldevorgang, damit diese die im vorhergehenden Abschnitt bereits angesprochenen Listen aktualisieren, welche die zu diesem Zeitpunkt existierenden Gruppen und deren zugehörige Nutzer beinhalten. Gehören nach dem Verlassen eines Nutzers keine weiteren Nutzer der entsprechenden Gruppe an, so wird auch diese Gruppe vom Sitzungsmanager entfernt. Der Nutzer bleibt beim Verlassen eines Dienstes weiterhin am System angemeldet.

Das Entfernen eines Nutzers aus einem Dienst durch den Administrator ist unter anderem dafür vorgesehen, falls sich der Nutzer nicht angemessen bei der Nutzung des Dienstes verhält. Hierbei wird vom zentralen Server eine entsprechende Steuernachricht versendet, welche beim betroffenen Nutzer den entsprechenden Dienst zur sofortigen Beendigung auffordert. Da im GWOTS der zentrale Client die einzelnen Dienst-Client-Module beinhaltet, muss dementsprechend diese Steuernachricht an diesen übermittelt werden.

Beim Verlassen des Systems durch den Nutzer beendet der zentrale Client alle noch eventuell aktiven Dienst-Client-Module, um die Konsistenz des Sitzungsmanagements sicherstellen zu können.

Außerdem besitzt der Administrator auch die Möglichkeit, einen angemeldeten Nutzer aus dem gesamten System zu entfernen. Hierzu wird analog zur Vorgehensweise zum Entfernen eines Nutzers aus einem Dienst eine entsprechende Steuernachricht vom zentralen Server an den betroffenen Client gesendet, nur dass diese den Client zum sofortigen Verlassen des Systems auffordert. Auch hier wird vom zentralen Client sichergestellt, dass zuvor noch eventuell aktive Dienst-Client-Module beendet werden.

3.1.2 Datenverwaltung

Im System GWOTS ist im zentralen Server ein grafisches Interface integriert, welches dem Administrator die Datenansicht und Manipulation der in der Datenbank befindlichen Nutzerdaten ermöglicht. Der dazu erforderliche Zugriff auf die gespeicherten Daten erfolgt dabei mit Hilfe der deskriptiven Standard-Datenbank-Sprache SQL, welche von der verwendeten Datenbank unterstützt wird. Außerdem werden in der Datenbank zusätzlich noch die Adressen der Startseiten zum Vorlesungsinhalt abgelegt, um es zu ermöglichen, dass einem Nutzer bei dessen Anmeldung vom zentralen Server die Startseite in Abhängigkeit vom zugehörigen Matrikel des Nutzers übermittelt werden kann.

Zur Ansicht der in der Datenbank vorhandenen Nutzerdaten wird eine hierarchische Gliederung verwendet, welche dem Administrator grafisch in Form einer Baumstruktur präsentiert wird. Eine solche Struktur bietet den Vorteil, dass man nach den Daten eines oder mehrerer Nutzer anhand verschiedener Kriterien, welche die oberste Ebene dieses Baumes bilden, suchen kann. Solche Kriterien sind im vorliegenden Fall die folgenden:

- realer Name eines Nutzers

- Kennung eines Nutzers
- Matrikel
- Fachbereich

Bei der Suche nach Nutzerdaten anhand des realen Namens beziehungsweise der Kennung werden alle in der Datenbank vorhandenen Nutzernamen beziehungsweise Nutzerkennungen in alphabetischer Reihenfolge innerhalb des entsprechenden Zweiges der Baumstruktur in einer untergeordneten Ebene aufgelistet. Die Anzeige der personenbezogenen Daten des gesuchten Nutzers erfolgt durch Auswahl des entsprechenden Namens beziehungsweise der zugehörigen Kennung.

Wird das Matrikel als Suchkriterium verwendet, so werden in der diesem Kriterium untergeordneten Ebene zunächst alle in der Datenbank registrierten Matrikel aufgelistet. Dies erfolgt ebenfalls in alphabetischer Reihenfolge. Durch Auswahl eines dieser Matrikel wird wiederum eine zu diesem Zweig untergeordnete Ebene geöffnet, welche die alphabetisch geordnete Auflistung aller Nutzernamen enthält, die zu diesem Matrikel gehören. Durch Auswahl eines dieser Nutzernamen werden dann die gesuchten Nutzerinformationen angezeigt.

Analog zur Anzeige der Nutzerdaten anhand des zugehörigen Matrikels gestaltet sich auch der Ablauf bei Verwendung des Fachbereiches als Suchkriterium. Der einzige Unterschied besteht darin, dass anstatt der vorhandenen Matrikel die in der Datenbank registrierten Fachbereiche aufgelistet werden.

In *Abbildung 3.4* wird die hier beschriebene hierarchische Strukturierung der Nutzerdaten anhand der vorab aufgezählten Suchkriterien grafisch dargestellt. Daraus wird auch ersichtlich, dass die Suchkriterien *Matrikel* und *Fachbereich* entsprechend der vorangegangenen Aussagen eine weitere hierarchische Gliederung beinhalten, welche die Nutzernamen nach Matrikeln beziehungsweise Fachbereichen getrennt auflistet. Die Blätter der dargestellten Baumstruktur bieten für alle dargestellten Verzweigungsmöglichkeiten den Zugriff auf die gesuchten Daten desjenigen Nutzers, welcher anhand des Nutzernamens beziehungsweise Logins auf Blattebene ausgewählt wird. Die grafische Darstellung des hier besprochenen Sachverhaltes zeigt einen beispielhaften Ausschnitt aus der Gesamtmenge der in der Datenbank vorhandenen Daten, wobei die gestrichelten Linien auf weitere Elemente innerhalb der ausgewählten Ebene der Baumstruktur verweisen.

Sämtliche Nutzerdaten müssen vom Administrator auch manipuliert werden können. Hierzu sind folgende Funktionalitäten vorgesehen:

- Anlegen, Verändern und Entfernen eines Fachbereiches in beziehungsweise aus der Datenbank
- Anlegen, Verändern und Entfernen eines Matrikels in beziehungsweise aus der Datenbank, wobei ein Matrikel immer genau einem Fachbereich zugeordnet wird

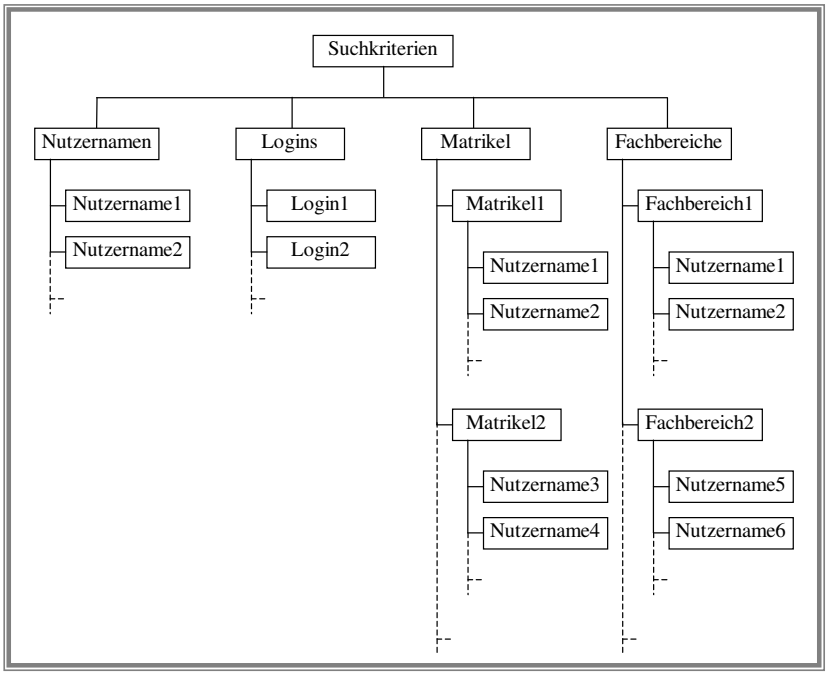

Abbildung 3.4: Hierarchische Struktur zur Ansicht der Nutzerdaten

- Anlegen, Verändern und Entfernen eines Nutzers in beziehungsweise aus der Datenbank, wobei ein Nutzer immer genau einem Matrikel zugeordnet wird.

Die Notwendigkeit der Gliederung von Operationen zur Datenmanipulation nach obigem Muster ergibt sich aus folgenden Aussagen, welche die Objekte der realen Welt auf das vorliegende Datenbankkonzept abbilden:

- zu einem Fachbereich gehören kein, ein oder mehrere Matrikel
- ein Matrikel gehört zu genau einem Fachbereich
- zu einem Matrikel gehören kein, ein oder mehrere Nutzer
- ein Nutzer gehört zu genau einem Matrikel.

Wie man hieraus erkennt, muss es dem Administrator also möglich sein, einen Fachbereich in der Datenbank anzulegen, ohne dass diesem Matrikel oder Nutzer angehören. Das Anlegen eines Matrikels hingegen erfordert die Existens des zugehörigen Fachbereiches, jedoch nicht das Vorhandensein von Nutzern. Beim Anlegen eines Nutzers werden jedoch sowohl die Existenz des übergeordneten Matrikels als auch indirekt das Vorhandensein des entsprechenden Fachbereiches im Datenbanksystem vorausgesetzt.

Während beim Entfernen eines Nutzers aus dem Datenbanksystem nur dessen Daten gelöscht werden, verhält es sich beim Entfernen eines Matrikels ganz anders. Da jeder Nutzer

genau einem Matrikel zugeordnet sein muss, werden beim Entfernen eines Matrikels zur Wahrung der Datenkonsistenz zusätzlich auch alle Nutzer entfernt, die diesem Matrikel angehören. Das Entfernen eines Fachbereichs aus der Datenbank bringt die gleichen Konsequenzen mit sich, da ein Matrikel ebenfalls genau einem Fachbereich zugeordnet wird. Hier werden also neben dem Fachbereich auch alle angehörenden Matrikel und wiederum deren zugehörige Nutzer aus dem Datenbankbestand gelöscht.

In der *Abbildung 3.5* wird das Konzept zur angesprochenen Datenbank in Form eines Entity-Relationship-Diagramms dargestellt, wobei dessen Aufbau und Komplexitäten auf den vorab getroffenen Aussagen beruhen. Aus Gründen der Übersichtlichkeit werden hierbei nicht alle Attribute des Entity *Nutzer* aufgeführt, welche aber im Folgenden noch in vollem Umfang beschrieben werden.

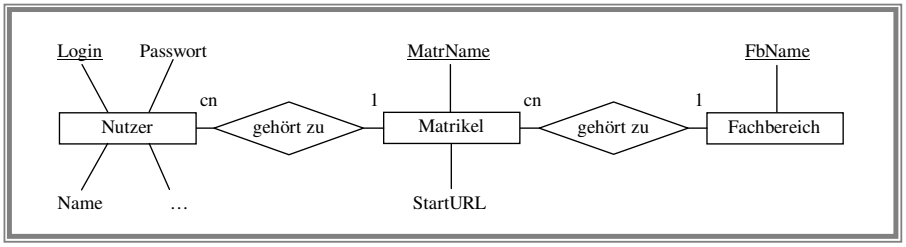

Abbildung 3.5: Entity-Relationship-Diagramm für die Benutzerverwaltung

Nach Anwendung der Transformationsregeln des RDM auf das in *Abbildung 3.5* abgebildete Entity-Relationship-Diagramm ergibt sich das folgende Relationenschema:

- 1:cn Beziehung zwischen *Matrikel* und *Nutzer*: Primärschlüssel *MatrName* aus Relation *Matrikel* wird als Fremdschlüssel in Relation *Nutzer* übernommen
- 1:cn Beziehung zwischen *Fachbereich* und *Matrikel*: Primärschlüssel *FbName* aus Relation *Fachbereich* wird als Fremdschlüssel in Relation *Matrikel* übernommen.

Es ist zu erwarten, dass die Primärschlüssel *FbName* und *MatrName* in den Relationen *Matrikel* beziehungsweise *Nutzer* häufig als Fremdschlüssel in den entsprechenden Relationen vorkommen werden. Da die Konzeption beinhaltet, dass der Administrator die genannten Attribute, die letztendlich den Namen des Fachbereichs beziehungsweise Matrikels beinhalten, jederzeit ändern kann, müssten im hier vorliegenden Fall bei einer solchen Aktualisierung auch die in den Relationen *Nutzer* beziehungsweise *Matrikel* korrespondierenden Fremdschlüssel aktualisiert werden. Um dem aus dem Wege zu gehen, wird in den Relationen *Fachbereich* und *Matrikel* ein zusätzliches Attribut eingeführt, welches einen „künstlichen“Index darstellt. Diese beiden Attribute *FbID* und *MatrID* bilden fortan den alleinigen Primärschlüssel für die Relationen *Fachbereich* beziehungsweise *Matrikel*. Der Vorteil einer solchen Vorgehensweise liegt darin, dass bei einer Änderung der Attribute *FbName* beziehungsweise *MatrName* der zu-

gehörige „künstliche“Primärschlüssel, welcher in den Relationen *Nutzer* beziehungsweise *Matrikel* als Fremdschlüssel dient, unberührt bleibt. Somit beschränkt sich eine Aktualisierung des Fachbereichnamens und Matrikelnamens ausschließlich auf die in den Relationen *Fachbereich* und *Matrikel* vorkommenden Attribute *FbName* beziehungsweise *MatrName*. Den Nachteil einer damit verbundenen Redundanz kann man hierfür in Kauf nehmen.

Die bei der Transformation des RDM entstandenen Relationen enthalten die folgenden Attribute, wobei die Primärschlüssel durch Unterstreichung gekennzeichnet werden:

- Nutzer(Login, Passwort, Name, Vorname, MatrID, MatrNr, Mail, Basis, Chat, Whiteboard)
- Matrikel(MatrID, MatrName, FbID, StartURL)
- Fachbereich(FbID, FbName).

Die in den hier vorgestellten Relationen vorkommenden Attribute besitzen dabei die folgende Bedeutung::

- *Login*: zur Systemanmeldung erforderliche eindeutige Kennung des Nutzers
- *Passwort*: zur Systemanmeldung erforderliches Passwort für den Nutzer
- *Name*: Nachname des Nutzers
- *Vorname*: Vorname des Nutzers
- *MatrNr*: Matrikelnummer des Nutzers
- *Mail*: eMail-Adresse des Nutzers
- *Basis*: Berechtigung zur Nutzung des Systems
- *Chat*: Berechtigung zur Nutzung des Chat-Dienstes
- *Whiteboard*: Berechtigung zur Nutzung des Whiteboard-Dienstes
- *MatrName*: Bezeichnung des Matrikel
- *StartURL*: bezeichnet die zu einem Matrikel zugehörige Adresse der Startseite
- *FbName*: Name des Fachbereiches.

Die Attribute zur Berechtigung der Nutzung des Systems oder des Chat- beziehungsweise Whiteboard-Dienstes ermöglichen es, den entsprechenden Dienst bei unangemessenem Verhalten des Nutzers zeitweilig oder permanent für diesen zu sperren. Diese Sperrung behält aufgrund der hierzu hinterlegten Daten in der Datenbank auch nach einem Neustart des zentralen Servers ihre Gültigkeit. Über das Attribut *Basis* lässt sich somit ein Benutzer-Account sperren, ohne diesen aus der Datenbank entfernen zu müssen.

3.2 Ressourcenabfrage und -auswertung

Beim Einsatz des hier vorgestellten Distance-Learning-Systems und den dabei zur Verfügung stehenden Diensten ergeben sich verschiedene Anforderungen an die Hardware-Konfiguration und Software-Ausstattung des Rechners, auf welchem dieses System ausgeführt wird. So sind beispielsweise unbedingt eine Netzanbindung des Rechners und das Vorhandensein des Internet Explorers erforderlich, um dieses Distance-Learning-System überhaupt nutzen zu können. Um einen ordnungsgemäßen Arbeitsablauf beim Einsatz des Distance-Learning-Systems garantieren zu können, ohne dabei dem Anwender das Prüfen der dazu notwendigen Voraussetzungen zu überlassen, wird auf dem Rechner des Nutzers vom zentralen Client bei dessen Start eine Ressourcen-Abfrage ausgeführt, deren Ergebnis bei der Anmeldung des Nutzers an den zentralen Server weitergeleitet und von diesem ausgewertet wird. Hierbei sind die folgenden Ressourcen-Informationen von besonderem Interesse:

- Netzwerk
- Prozessor
- Arbeitsspeicher
- Software-Komponenten.

Die hierbei ermittelten Informationen über die Netzanbindung des Rechners bezüglich der verfügbaren Bandbreite sind von entscheidender Bedeutung für die Übermittlung von Videodaten jeglicher Art. Hiernach richtet sich diesbezüglich beispielsweise auch die maximal verfügbare Bitrate für den im System integrierten Videokonferenz-Dienst beziehungsweise dessen Verfügbarkeit überhaupt. Auch die verfügbare Prozessorleistung und Größe des Arbeitsspeichers sind ein entscheidendes Kriterium bei der Übertragung von audiovisuellen Informationen, da das im vorliegenden Entwurf enthaltene Konzept einen Dienst zur Videokommunikation beinhaltet, der auch ohne spezielle Videoschnittkarte einen solchen Dienst zur Verfügung stellt. Das bedeutet, dass die eintreffenden Videodaten per Software-Codec dekodiert werden müssen, was eine entsprechende Belastung des Prozessors der Client-Maschine nach sich zieht und zur schnellstmöglichen Ausführung der dabei anfallenden Berechnungen außerdem entsprechend Arbeitsspeicher voraussetzt. Die Informationen über installierte Softwarekomponenten sind für den eMail-Dienst von Bedeutung, da dieser den auf dem jeweiligen System installierten Standard-Mail-Client verwendet. Eine fehlende Installation eines Mail-Clients auf dem Rechner des Nutzers hat dementsprechend zur Folge, dass der im Distance-Learning-System integrierte eMail-Dienst dem Anwender nicht zur Verfügung gestellt werden kann.

Die Informationen über die auf dem jeweiligen Rechnersytem verfügbaren Ressourcen sind unter Microsoft Windows^{TM}9x/Me/NT/2000/XP über die Registry-Datenbank abrufbar.

Für das hier vorgestellte System werden die folgenden Mindestanforderungen an die Hardware-Konfiguration und Software-Ausstattung des Client-Rechners gestellt, um nach Möglichkeit alle verfügbaren Dienste nutzen zu können:

- Pentium 233 MHz (bzw. vergleichbare CPU anderer Hersteller) oder höher
- 64 MB Hauptspeicher oder mehr
- Internet Explorer ab Version 4
- eMail-Client (z.B. Outlook Express)
- Windows^{TM}9x/Me/NT/2000/XP
- Netzwerk mit einer Bandbreite von 56.6 Kbit/s oder mehr.

In Abhängigkeit von der Erfüllung der aufgeführten Anforderungen sind die Dienste beim Anwender verfügbar, oder auch nicht. Der Dienst zur Audio/Video-Kommunikation beispielsweise ist für eine Netzanbindung mit einer Bandbreite von weniger als 56.6 Kbit/s eingeschränkt nutzbar, was bedeutet, dass in einem solchen Fall die Übertragung von Videodaten deaktiviert wird, der Anwender aber dennoch an einer solchen Konferenz teilnehmen und dabei zumindest per Sprachübertragung mit den anderen Teilnehmern kommunizieren kann.

Jede im GWOTS integrierte Interaktionskomponente stellt eine Auflistung der von ihr gestellten Hardware- und Software-Anforderungen zusammen und dem zentralen Server zur Verfügung. Dieser kann anhand der jeweiligen Listen entscheiden, ob für den zugehörigen Dienst auf der Client-Maschine die geforderten Ressourcen verfügbar sind und dementsprechend reagieren, indem er den Dienst freigibt, oder nicht.

3.3 Interaktionskomponenten

Im Rahmen dieses Kapitels werden die Konzepte für die im GWOTS enthaltenen Interaktionskomponenten vorgestellt. Die Dienste zur Nutzerkommunikation via Chat und Whiteboard werden dabei aus den im *Kapitel 3* besagten Gründen in Form von Modulen entworfen, welche zur Nutzer- und Gruppenverwaltung den Sitzungsmanager des zentralen Servers verwenden und deshalb fest in das Server-Modul des zentralen Servers integriert werden. Der Dienst zur Videokonferenz bietet zudem im GWOTS die Möglichkeit der Bildung von Konferenzgruppen, wobei eine breitbandige Netzanbindung für die Nutzung dieses Dienstes nicht erforderlich ist. Außerdem wird in der nachfolgend beschriebenen Konzeption zur Präsentation von Vorlesungsinhalten auch die verfügbare Bandbreite der Netzanbindungen einzelner Nutzer berücksichtigt, um diesen Dienst, welcher aufgrund der in den Lehrseiten integrierten Videosequenzen ein entsprechend hohes Datenaufkommen beim Nutzer verursacht, für jeden Teilnehmer auch bei Nutzung einer schmalbandigen Netzanbindung nutzbar zu machen.

Alle im GWOTS enthaltenen Interaktionskomponenten weisen die Gemeinsamkeit auf, dass sie Daten über Netzwerkverbindungen verschicken und empfangen. Bei den Diensten zur Nutzerkommunikation via Chat, Whiteboard oder Videokonferenz ist es außerdem so, dass das zugehörige Dienst-Server-Modul ein von ihm empfangenes Datenpaket an alle Mitglieder derjenigen Gruppe weiterleiten muss, welcher der Absender des Paketes angehört. Multicasting wurde speziell für eine derartige Datenverteilung entwickelt, um die Ressourcen des verteilenden Servers zu schonen. In der Praxis ist es aber leider so, dass ein Großteil des Internets nicht Multicast-fähig ist [KUE01]. Um aber unabhängig von der Verfügbarkeit eines Multicast-fähigen Netzes den Einsatz des hier entwickelten Systems gewährleisten zu können, verwenden die betreffenden Interaktionskomponenten, welche im Rahmen dieser Arbeit konzipiert werden, deshalb Unicasting zur Datenverteilung.

3.3.1 Dienst zur Präsentation der Vorlesungsinhalte

Der Dienst zur Präsentation der Vorlesungsinhalte stellt in einem Distance-Learning-System die Hauptanwendung dar, wobei er in der Lage sein muss, textuelle, grafische und audiovisuelle Informationen unterschiedlichster Datenformate an den Anwender zu übermitteln und bei diesem auf grafische und akustische Weise darzustellen. Bei den dabei angebotenen audiovisuellen Informationen handelt es sich um sogenannte OnDemand-Inhalte, was bedeutet, dass es sich um entsprechend vorbereitete Dateien handelt, die vom Anwender jederzeit zur Ansicht angefordert werden können [KUE01]. Zur Umsetzung der hierbei erforderlichen Funktionalitäten wird die Dokumentbeschreibungssprache HTML verwendet, wobei zur Verwaltung der Dokumente, welche die Vorlesungsinhalte in digitaler Form beinhalten, ein Web-Server eingesetzt wird. Die Anzeige dieser Dokumente erfolgt beim Anwender mit Hilfe eines Web-Browsers, welcher fest in die Benutzeroberfläche des zentralen Clients integriert ist. Die Vorteile dieses Konzeptes sind die folgenden:

- hohe Flexibilität dadurch, dass ein Web-Browser alle relevanten Datentypen darstellen kann
- Zukunftssicherheit dadurch, dass Web-Browser ständig um neue Funktionalität erweitert werden
- große Auswahl an Lehrmaterialien, die bereits in Form von HTML-Dokumenten vorliegen
- Existenz von geeigneten HTML-Editoren, mit denen neue Inhalte erstellt werden können
- geringer Implementierungsaufwand für die Darstellung der Inhalte auf der Client-Seite.

In [ROS00] wird aufgrund durchgeführter Tests darauf verwiesen, dass es nicht ausreicht, allein die Videoaufnahmen der Lehrveranstaltung zu übertragen. Dies begründet sich dadurch,

dass infolge der verfügbaren Bandbreite von Netzanbindungen die Videoaufnahmen zur Übertragung in komprimierten Formaten und mit begrenzter Auflösung abgespeichert werden müssen und somit die Qualität der ursprünglichen Videoaufnahme gemindert wird. Diese Qualitätsminderung nimmt dabei mit steigender Komprimierung und sinkender Auflösung zu. In den Videoaufnahmen vorkommende Details, wie beispielsweise die vom Vortragenden auf eine Wandtafel geschriebenen Texte oder Diagramme, sind somit bei der Wiedergabe einer solchen Videoaufnahme aus besagtem Grund nicht mehr lesbar. Deshalb macht es sich erforderlich, dass synchron zur Wiedergabe der Audio/Video-Aufnahme, welche den Lehrenden bei seinen Ausführungen zur Vorlesung zeigt, auch grafische und alphanumerische Informationen an den Anwender übertragen und diesem präsentiert werden. Zu diesem Zweck wird die Videoaufzeichnung einer Lehrveranstaltung in mehrere Fragmente zerlegt, um dem Anwender das Navigieren zwischen einzelnen Vorlesungsabschnitten zu gestatten. Hierbei wird immer zum aktuell betrachteten Dokument das zugehörige Videofragment abgespielt.

Aufgrund dieser Tatsache werden im hier vorliegenden Konzept die Videoaufzeichnungen der Vorlesungen in Fragmente zerlegt, welche dem Anwender synchron mit den jeweils zugehörigen Informationen grafischer und alphanumerischer Form präsentiert werden. Hierzu wird das Dokument in zwei separate Teile, also zwei HTML-Frames, untergliedert. Während der erste Teil die Videoaufzeichnung über ein entsprechendes Browser-Plugin einbindet und wiedergibt, werden im zweiten Teil die hierzu begleitenden Informationen, wie etwa Tabellen, Grafiken oder stichpunktartige Aufzeichnungen, angezeigt. Diese Aufteilung des Dokumentes in zwei Teile ist erforderlich, da der begleitende Inhalt vom Umfang her eventuell den sichtbaren Bereich des Dokumentes im Browser-Fenster überschreitet, was zur Folge hat, dass der Anwender innerhalb dieses Dokumentes scrollen muss. Die Darstellung der Videoaufzeichnung bleibt hiervon jedoch unberührt und ist für den Anwender auch weiterhin sichtbar, da sie, wie beschrieben, in einem separaten Teil des Dokumentes angezeigt wird. Die hier beschriebene Aufteilung des HTML-Dokumentes wird in *Abbildung 3.6* dargestellt.

Die Aufteilung des in *Abbildung 3.6* dargestellten HTML-Dokumentes erfolgt durch eine vertikale Trennung. Während der linke Frame des Dokumentes hierbei die über das Browser-Plugin eingebundenen Audio/Video-Daten beinhaltet, dient der rechte Frame zur Darstellung der begleitenden Informationen. In der Abbildung wird auch am rechten Rand des HTML-Dokumentes der Scroll-Balken dargestellt, welcher das Scrollen innerhalb des rechten Frames erlaubt. Auf den linken Frame hat der Scroll-Balken dabei keinerlei Einfluss. Die im linken Frame eingebundenen Audio/Video-Inhalte können mit Hilfe der in *Abbildung 3.6* dargestellten und vom Plugin bereitgestellten Schaltflächen gesteuert werden. Mögliche Steueroperationen sind dabei das zeitweilige Anhalten, Beenden oder erneute Starten der Videoaufzeichnung.

Beim System OVID werden die erwähnten Videofragmente im MPEG-Format in einer Auflösung von 352 * 288 Pixeln und einer Bildwiederholrate von 25 fps abgespeichert [ROS00]. Dem Anwender werden sie dann zusammen mit den HTML-Seiten der Vorlesungsinhalte über

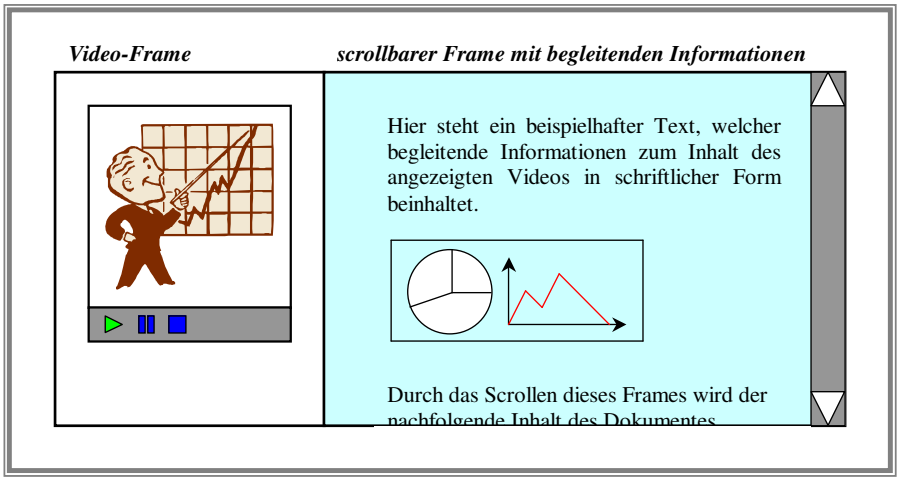

Abbildung 3.6: Gliederung des HTML-Dokumentes zur Lehrstoffpräsentation in zwei separate Teile

den zur Dokumentenverwaltung verwendeten Web-Server übermittelt. Der dabei anfallende Datenstrom im Umfang von 1.2 Mbit/s hat während Testläufen des OVID gezeigt, dass es bei einer zur Verfügung stehenden Bandbreite seitens des Nutzers von 115200 bit/s (in etwa die verfügbare Bandbreite einer ISDN-Kanalbündelung) länger als eine Minute dauert, um ein derartiges Videofragment mit einer Gesamtspieldauer von 6 Sekunden beim Anwender im Browser-Fenster darzustellen [ROS00], wobei neben der damit verbundenen Verzögerung der Darstellung außerdem noch Aussetzer bei der Wiedergabe auftreten.

Aufgrund der Tatsachen, dass gerade der hier beschriebene Dienst die Hauptanwendung im Distance-Learning-System darstellt und nicht jeder Anwender über eine breitbandige Netzanbindung verfügt, wird im vorliegenden Konzept zu diesem Dienst eine Lösung aufgezeigt, die auch die Verwendung schmalbandiger Netzanbindungen berücksichtigt. Um also auch Anwendern die Nutzung des Distance-Learning-Dienstes über eine Modemverbindung zu ermöglichen, müssen die Videofragmente bezüglich ihrer Größe, welche von der Auflösung, der Bildwiederholrate und der Komprimierung des Videos bestimmt wird, an die Gegebenheiten der verfügbaren Bandbreite beim Nutzer angepasst werden. Gleiches gilt auch für die darin enthaltenen Audiodaten. Eine solche Anpassung bringt Verluste bezüglich der Qualität der Aufzeichnung mit sich, ist aber unter dem Gesichtspunkt der Verfügbarkeit des Dienstes für alle Nutzer unumgänglich. Um aber einem Nutzer, der über eine breitbandige Netzanbindung verfügt, nicht unnötigerweise Videoinformationen in für seine Verhältnisse minderwertiger Qualität zu übertragen, müssen die Videoaufzeichnungen für gleiche Inhalte in mehreren Qualitätsstufen verfügbar sein. Die Übermittlung des Videos zu einem bestimmten Lehrinhalt kann somit in Abhängigkeit von der verfügbaren Bandbreite erfolgen. Die empfohlenen Auflösungen und Bildwiederholfrequenzen für typische Bandbreiten werden in *Tabelle 3.1* aufgeführt.

Zielgruppe	Durchschnittlich verfügbare Bandbreite	Typ. Audioanteil	Typ. Videogröße und Bildfrequenz
28.8 Kbps Modem	20 Kbps	6 Kbps	176x132, 7fps
56.6 Kbps Modem	34 Kbps	8 Kbps	176x132, 10 fps
64 Kbps ISDN	45 Kbps	11 Kbps	176x132, 15 fps
128 Kbps Dual-ISDN	80 Kbps	20 Kbps	240x180, 15 fps
DSL	300 Kbps	64 Kbps	320x200, 25 fps

Tabelle 3.1: Typische Bandbreiten von Internetverbindungen [KUE01]

Zur Bereitstellung der in Dateiform vorhandenen Videoaufzeichnungen kommen sowohl ein Web-Server als auch ein Streaming-Server in Betracht. Da die Videoaufzeichnungen laut Konzeption in von der verfügbaren Bandbreite abhängigen Qualitätsstufen angeboten werden, muss bei Verwendung eines Web-Servers die entsprechende Auswahl entweder durch den Anwender erfolgen, oder aber vom Distance-Learning-System übernommen werden. Beides bringt Nachteile mit sich. Während die entsprechende Auswahl durch den Nutzer die vom Konzept geforderte einfache Handhabung für ungeübte Anwender in Frage stellt, da diese hierzu zumindest gewisse Kenntnisse über die von ihnen verwendete Netzanbindung besitzen müssten, zieht die Auswahl durch das System einen erheblichen Implementierungsaufwand nach sich. Deshalb werden im GWOTS sowohl ein Web-Server als auch ein Streaming-Server eingesetzt, wobei der Web-Server für die Verwaltung der HTML-Dokumente verantwortlich ist und der Streaming-Server zur Verteilung der Video-Daten dient (vgl. *Abbildung 3.7*). Der hier angesprochene Streaming-Server besitzt die Eigenheit, dass er bei einer Anforderung für eine Videodatei durch das im HTML-Dokument eingebundene Plugin über das hierbei verwendete Protokoll (z.B. das *Microsoft Media Server Protocol* bei Verwendung des *Windows Media Servers*^{TM}) zunächst mit diesem Informationen austauscht, welche Angaben über die verfügbare Bandbreite des Empfängers beinhalten. Somit kann der Streaming-Server entscheiden, welche der bereitstehenden Qualitätsstufen für die Übertragung der Audio/Video-Daten verwendet werden kann [KUE01]. Dabei wird neben Bildqualität, Framerate und Auflösung der Videodaten auch die Bitrate der enthaltenen Audiodaten berücksichtigt [WME02].

Ein weiterer Vorteil des Einsatzes eines Streaming-Servers besteht darin, dass dieser auch während der Datenübertragung Änderungen der verfügbaren Bandbreite erkennen und dementsprechend flexibel in die nächst höhere oder niedrigere Qualitätsstufe wechseln kann. Das hat den Vorteil, dass die Präsentation zum einen möglichst optimal beim Anwender dargestellt wird, und zum anderen die Wiedergabe nicht abreißt, weil eventuell Daten nachgeladen werden müssten [KUE01].

Die *Abbildung 3.7* verdeutlicht auf grafische Weise die Zuständigkeit des Web-Servers beziehungsweise Streaming-Servers für die dargestellten Lehrinhalte sowie den dabei auftretenden Datenaustausch. Hierzu wird neben diesen beiden Servern auch der zentrale Client dargestellt, welcher zur Präsentation der Vorlesungsinhalte über einen integrierten Web-Browser verfügt. Das in der Abbildung enthaltene Element mit der Bezeichnung *Video* bezeichent hierbei das in die HTML-Seite integrierte Browser-Plugin, welches die Videodatei vom Streaming-Server anfordert und dem Anwender zur Ansicht bringt.

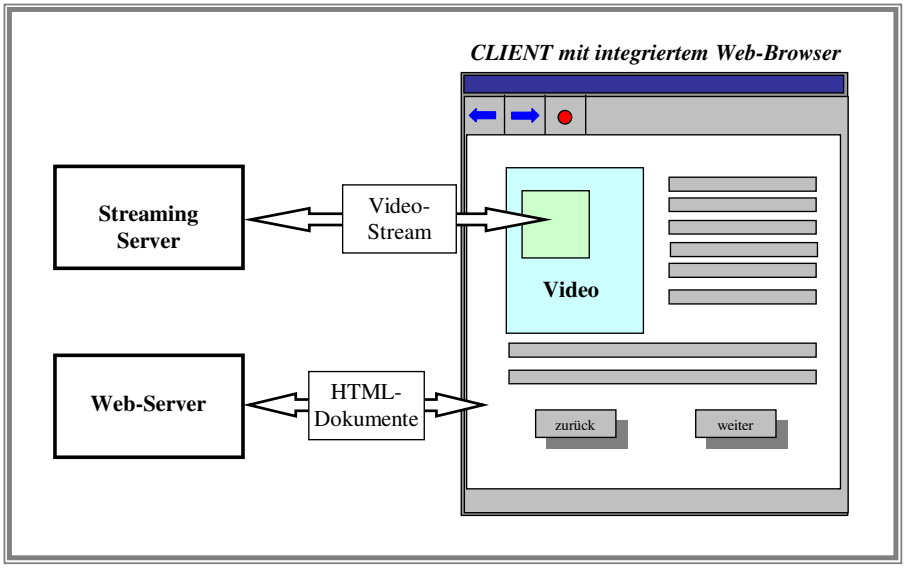

Abbildung 3.7: Datenaustausch zwischen Client und Web-Server bzw. Streaming-Server

Zur Generierung der Videodateien in verschiedenen Qualitätsstufen wird die ursprüngliche Videoaufzeichnung mit einem speziellen Werkzeug, beispielsweise dem *Windows Media Encoder*^{TM} bei Verwendung des *Windows Media Servers*^{TM}, in das vom Streaming-Server benötigte Datenformat umgewandelt. Bei der hierbei generierten Datei handelt es sich um eine Multistream-Datei, was bedeutet, dass diese mehrere Streams enthält, wobei jeder dieser Streams den gleichen Inhalt umfasst, aber eben in unterschiedlichen Qualitätsstufen. Die Anzahl der Qualitätsstufen und deren einzelne Parameter, also Bildqualität, Framerate, Auflösung und Audio-Bitrate, werden vor der Generierung der Multistream-Datei festgelegt und richten sich dabei nach den verfügbaren Bandbreiten der Zielgruppe. *Tabelle 3.1* gibt hierfür Empfehlungen an.

3.3.2 Chat

Der Chat-Dienst wird im hier entwickelten System fest in den zentralen Client beziehungsweise den zentralen Server integriert. Der Aufgabenbereich dieses Dienstes besteht dabei im Austausch von Textnachrichten, wobei eine solche Nachricht eines zu einer Chat-Gruppe ge-

hörenden Anwenders an alle anderen Mitglieder dieser Gruppe übertragen werden muss.

Um den Aufgabenbereich eines Chat-Dienstes umsetzen zu können, basiert ein Chat-System ebenfalls auf einer Client/Server-Architektur. Während die daran beteiligten Chat-Clients der Eingabe und Ausgabe der vom Nutzer verfassten Textnachrichten dienen, ist der Chat-Server für den verwaltungstechnischen Bereich der Verteilung dieser Nachrichten an die entsprechenden Mitglieder einer Chat-Gruppe verantwortlich. Zur Verwaltung der an den Chat-Dienst angemeldeten Gruppen und deren zugehörigen Nutzern wird der Sitzungsmanager des zentralen Servers verwendet.

Der für das System GWOTS entwickelte Chat-Dienst beinhaltet alle Funktionalitäten, die dem Nutzer den notwendigen Komfort bei dessen Benutzung bieten und somit von solch einem System erwartet werden. Im Einzelnen werden dabei folgende Anforderungen erfüllt:

- Möglichkeit der Verwaltung mehrerer Chat-Gruppen
- Nutzer hat die Möglichkeit, eine Chat-Gruppe zu erstellen beziehungsweise einer bereits bestehenden beizutreten
- Nutzer erhält bereits vor dem Eintritt in eine Chat-Gruppe eine Auflistung der an der dort geführten Diskussion beteiligten Teilnehmer
- zur besseren Lesbarkeit der Beiträge einer Gruppe werden diese, differenziert nach ihren Verfassern, durch verschiedene Farben dargestellt.

Der in dieser Aufzählung aufgeführte dritte Punkt ermöglicht es dem Nutzer, neben der Auswahl einer Chat-Gruppe anhand des Diskussionsthemas, welches in der Regel aus der Gruppenbezeichnung hervorgeht, auch gezielt nach anderen Chat-Teilnehmern suchen zu können, um sich an deren Diskussion zu beteiligen.

Das beim Chat-Dienst anfallende Datenaufkommen ist in Hinsicht auf die dafür beim Server aufkommende Netzlast von großer Bedeutung. Dies wird dadurch begründet, dass bei der Verwendung von Unicast zur Datenverteilung einer Nachricht diese vom Server an jeden Teilnehmer der Gruppe einzeln versendet wird. Je mehr Teilnehmer einer Gruppe angehören, desto größer wird die dabei anfallende Datenmenge pro Nachricht. In der Konzeption von [ROS00] werden für den dort eingesetzten Chat-Dienst die in *Tabelle 3.2* enthaltenen Werte angegeben. Als Grundlage zur dazu durchgeführten Berechnung wird veranschlagt, dass ein Benutzer maximal 5 Zeichen pro Sekunde eingeben kann und 100 Benutzer den Chat-Dienst nutzen. Außerdem wird hierbei festgelegt, dass ein Zeichen mit 8 bit kodiert wird. Daraus ergibt sich, dass jeder Teilnehmer des Chats eine Datenmenge von maximal 40 bit/s an den Chat-Server versendet, der diese dann an alle zur entspechenden Gruppe gehörenden Teilnehmer weiterleiten muss. Für das aufgeführte Beispiel für 5 Gruppen mit je 20 Teilnehmern ergibt sich also für die Eingaben eines einzelnen Nutzers eine Datenmenge von *5 Zeichen/s * 20 Teilnehmer * 8 Bit/Zeichen = 0,8 Kbit/s*, die der Chat-Server insgesamt pro Gruppe zu verteilen hat. Schreiben

dagegen alle Teilnehmer einer Gruppe gleichzeitig, so ergibt sich eine Datenmenge von *0,8 Kbit/s * 20 = 16 Kbit/s* pro Gruppe. Für alle 5 Gruppen sind das demnach also *16 Kbit/s * 5 = 80 Kbit/s*, die der Chat-Server an die Chat-Clients versenden muss.

Angaben	**100 Benutzer, 1 Gruppe**	**100 Benutzer, 5 Gruppen**
Schreibgeschwindigkeit eines Benutzers	5 Zeichen/s	5 Zeichen/s
Anzahl der Benutzer in einer Gruppe	100	20
Belastung des Netzwerks durch 1 Benutzer	4 Kbit/s	0.8 Kbit/s
Datendurchsatz pro Gruppe, maximal	400 Kbit/s	16 Kbit/s
Gesamte Belastung des Netzwerks, maximal	400 Kbit/s	80 Kbit/s

Tabelle 3.2: Datendurchsatz eines Chat-Server-Moduls (Nutzdaten ohne Overhead) [ROS00]

Aus den von [ROS00] durchgeführten Berechnungen wird klar ersichtlich, dass die Größe und Anzahl der Gruppen bei gleicher Gesamtanzahl von Teilnehmern in Bezug auf die anfallenden Datenmengen eine große Rolle spielt. Da in der Praxis Gruppen mit mehr als 20 Teilnehmer keine sinnvolle Diskussion mehr erlauben, sind die berechneten Werte einer Aufteilung von 100 Nutzern auf 5 Gruppen als realistisch zu betrachten. Die für eine Aufteilung von 100 Nutzern auf nur eine Gruppe berechneten Werte sind dagegen ein theoretischer Maximalwert unter den gegebenen Bedingungen.

Auffällig bei der hier vorgestellten Berechnung ist aber, dass dabei nur die reinen Nutzdaten einfliessen. Der nicht betrachtete Anteil des zum Datenversand einer Nachricht verwendeten Headers ist aber erheblich, da dieser die Kennung des Verfassers und die Angabe der Schriftfarbe enthält, mit welcher die Nachricht bei den Empfängern dargestellt wird. Die zuvor angewendete Rechnung kann unter Berücksichtigung der Header-Daten aber nicht mehr angewendet werden, da der Header zu einer Nachricht gehört, die in der Praxis in der Regel mehr als nur ein einziges Zeichen umfasst. Je größer die Nachricht also ist, desto geringer wirkt sich die zusätzliche Datenmenge des Headers auf die vom Chat-Server zu verteilende Datenmenge aus. Trotzdem bietet es sich an, nach Lösungsansätzen zur Optimierung der Header-Daten zu suchen. Zielstellung einer solchen Lösung muss neben einer Optimierung der anfallenden Header-Daten auch der Erhalt der Funktionalität des Ausgangssystems sein. Mit anderen Worten bedeutet dies, dass auch weiterhin der Verfasser der Nachricht in Verbindung mit dieser bei den Teilnehmern des Chats angezeigt wird und auch die farbliche Trennung der Nachrichten

nach ihren Verfassern erhalten bleibt.

Als Lösungsansatz für die Zuordnung einer Nachricht zu ihrem Verfasser wird im vorliegenden Entwurf eine ID in Form einer Zahl verwendet, die dem Nutzer beim Eintritt in eine Chat-Gruppe vom Sitzungsmanager zugewiesen wird und ihn innerhalb dieser Gruppe eindeutig identifiziert. Das Dienst-Server-Modul ist außerdem auch in der Lage, den Verfasser der Nachricht ohne den damit verbundenen Header zu identifizieren, da er die Nutzerkennung anhand des Chat-Socket der eingehenden Nachricht ermittelt. Die Anzeige der Nutzerkennung des Verfassers in Verbindung mit der zugehörigen Nachricht erfolgt auf Client-Seite unter Zuhilfenahme einer der in *Kapitel 3.1.1* vorgestellten Listen, welche die Nutzerkennungen und zugehörigen IDs aller in der Chat-Gruppe befindlichen Teilnehmer enthält.

Die vom Sitzungsmanager vergebene eindeutige ID wird hierbei durch eine fortlaufende Numerierung realisiert. Hierzu wird diese ID, welche durch einen ganzzahligen, nicht vorzeichenbehafteten Datentyp repräsentiert wird, beim Eröffnen der betreffenden Chat-Gruppe mit einem festgelegten Wert initialisiert, der dann beim Eintritt eines Nutzers in diese Gruppe inkrementiert und diesem zugewiesen wird. Wird beim Inkrementieren der ID der maximale Wertebereich überschritten, so beginnt die Zählung wieder vom kleinstmöglichen Wert an. Deshalb muss der Datentyp in ausreichender Größe gewählt beziehungsweise die Anzahl der Nutzer pro Gruppe entsprechend begrenzt werden. Da man in der Praxis davon ausgehen kann, dass einer Gruppe nicht mehr als 256 Teilnehmer beitreten werden, wird die Anzahl der möglichen Teilnehmer im vorliegenden Entwurf auf diesen Maximalwert eingeschränkt. Somit genügen bereits 8 bit, um diese Nutzer innerhalb ihrer Gruppe eindeutig zu kennzeichnen.

Unter der Voraussetzung, dass nach Möglichkeit die Nachrichten jedes Teilnehmers in einer anderen Farbe angezeigt werden, macht es eigentlich keinen Sinn, dass ein Nutzer beim Eintritt in den Chat diese Farbe selbst wählt, da ja auf diese Weise nicht sichergestellt werden kann, dass diese nicht bereits von einem anderen Teilnehmer verwendet wird. Deshalb übernimmt diese Auswahl bei der hier beschriebenen Chat-Komponente das System. Die farbliche Zuordnung einer Nachricht zu ihrem Verfasser auf Client-Seite erfolgt unter Zuhilfenahme der ihm zur Verfügung stehenden Liste, welche alle Kennungen der zur Gruppe gehörenden Teilnehmer enthält. Bei einer eingehenden Nachricht wird der Index der Teilnehmerliste, den man anhand der mit der Nachricht erhaltenen ID erhält, verwendet, um aus einer definierten Farbtabelle die entsprechende Farbe auszuwählen, mit welcher die Nachrichten eines Verfassers angezeigt werden. Eine Übermittlung der Farbe über den Header entfällt damit.

Mit dem hier vorgelegten Konzept wird der Header der Chat-Nachricht auf eine Größe von 8 bit optimiert, ohne dabei die Funktionalität des Dienstes einzuschränken. Selbst unter Einbeziehung dieses Headers in die von [ROS00] verwendete Berechnungsformel ergeben sich damit Werte, die für die Netzanbindung des Servers in Verbindung mit dem anfallenden Datenaufkommen beherrschbar sind.

Abweichend von den hier aufgeführten Berechnungen liegt das tatsächliche Datenaufkom-

men über dem hier berechneten, da vom TCP- und vom IP-Protokoll zur Datenübermittlung neben den Nutzdaten, also im vorliegenden Fall den Chat-Informationen, weitere Header-Daten in Form von Protokollinformationen übermittelt werden. Diese Header-Daten werden in die Berechnungen aber nicht mit einbezogen, da ihre Größe von verschiedenen Parametern abhängig ist.

3.3.3 Audio- und Videokommunikation

Der hier vorgestellte und in das System integrierte Dienst zur Nutzerkommunikation via Videokonferenz verfolgt das Ziel, dem Anwender eine Videokommunikationskomponente zu Verfügung zu stellen, welche in der Praxis für eine breite Masse von Nutzern einsetzbar ist. Das bedeutet, dass diese Komponente keine spezielle Hardware in Form von Videoschnittkarten voraussetzt und außerdem auch bei Verwendung von schmalbandigen Netzanbindungen benutzt werden kann. Für schmalbandige Netzanbindungen muss der bei der Übertragung von Audio/Video-Informationen auftretende Datenstrom entsprechend der verfügbaren Bandbreite begrenzt sein. Das hat zur Folge, dass die Komprimierung entsprechend hoch und die Auflösung entsprechend niedrig sein muss, was eine verhältnismäßig geringe Übertragungsqualität nach sich zieht. Die Qualität der übertragenen Audio/Video-Informationen steht aber in Hinblick auf die Verfügbarkeit des Dienstes für schmalbandige Netzanbindungen im Hintergrund.

Bei der Auswahl einer hierfür geeigneten Technologie erfolgt die Orientierung an bereits existierenden Kommunikationssystemen, welche in der Praxis angewendet werden und die geforderten Eigenschaften aufweisen. Hierzu gehören Videokonferenzprogramme wie Microsoft NetMeeting^{TM}, Cu-SeeMe oder OhPhone. Diese haben alle gemein, dass sie das von der ITU für die Videokommunikation via Internet verabschiedete H.323 Standardprotokoll zur Echtzeit-Übertragung der hierbei anfallenden Informationen verwenden. Dieses Protokoll erlaubt sowohl die Punkt-zu-Punkt-Verbindung zwischen zwei Kommunikationpartnern als auch eine Gruppenkommunikation von drei oder mehr Teilnehmern bei Verwendung einer MCU [GUL98], wobei diese einen Konferenz-Server darstellt. Außerdem hat der Anwender unter anderem auch die Möglichkeit der Auswahl unterschiedlicher Bitraten und Qualitätsstufen für die zu übertragenden Audio/Video-Daten [MIC02], was den Anforderungen der verfügbaren Bandbreiten Rechnung trägt.

Zum Kommunikationsaufbau zu einem am System angemeldeten Nutzer steht dem Anwender im hier konzipierten Dienst eine Liste mit deren Nutzerkennungen zur Auswahl, welche schon im Rahmen der zuvor vorgestellten Konzeption zum Sitzungsmanagement näher beschrieben wurde. Da bei einer solchen privaten Konferenz der Datenaustausch ohne zusätzlichen Server direkt zwischen den Konferenzkomponenten der beiden Teilnehmer erfolgt, beschränkt sich die Kommunikation zwischen zentralem Client und zentralem Server lediglich

auf den Austausch von Steuernachrichten, um das Konzept des einheitlichen Sitzungsmanagements auch für diesen Dienst umsetzen zu können. Der zentrale Server übermittelt dem anfragenden Client hierauf die IP-Adresse des gewünschten Kommunikationspartners, falls der Konferenzdienst für diesen freigegeben ist. Ist dies nicht der Fall, wird stattdessen die Nichtverfügbarkeit des Dienstes für den Zielrechner mitgeteilt. Analog hierzu erfolgt eine entsprechende Benachrichtigung des zentralen Servers durch die beiden Konferenzteilnehmer, wenn diese ihr Gespräch beenden.

Um die Videokonferenz in Form einer Gruppe umsetzen zu können, wird dem zentralen Client nach dessen Anmeldung unter anderem auch die IP-Adresse des Konferenz-Servers übermittelt (vgl. Seite 21). Zudem stehen dem Anwender für den Konferenzdienst noch Listen zur Verfügung, welche analog zu den Gruppen- und Mitgliederlisten des Chat-Dienstes auch hier den notwendigen Überblick gestatten. Nach Auswahl einer Gruppe aus der entsprechenden Liste wird eine Verbindung zum Konferenz-Server aufgebaut und die Anmeldung an die gewünschte Gruppe vollzogen. Um auch hier das einheitliche Sitzungsmanagement zu wahren, werden die hierbei durchgeführten Vorgänge zur An- und Abmeldung dem zentralen Server über entsprechende Steuernachrichten mitgeteilt. Die Kommunikationsstruktur hierzu wird aus *Abbildung 3.3* ersichtlich.

3.3.4 Whiteboard

Die Zielstellung des Konzeptes des im hier vorliegenden System integrierten Whiteboards ist es, dem Anwender ein Werkzeug zur Bearbeitung eines grafischen Dokumentes zur Verfügung zu stellen. Dieses ermöglicht dabei die gleichzeitige Bearbeitung eines solchen Dokumentes durch mehrere an das System angemeldete und zu einer Gruppe gehörigen Nutzer und stellt dabei die folgenden Funktionalitäten eines einfachen Bildbearbeitungsprogrammes bereit:

- **Linie**: Zeichnen von Linien in verschiedenen Strichstärken, Linienarten und Farben
- **Kreis/Ellipse**: Zeichnen von Kreisen beziehungsweise Ellipsen mit verschiedenen Füllfarben/-mustern sowie Linienstärken/-arten für den Umriss des Objektes
- **Rechteck**: Zeichnen von Rechtecken mit verschiedenen Füllfarben/-mustern sowie Linienstärken/-arten für den Umriss des Objektes
- **Rechteck mit abgerundeten Ecken**: Zeichnen von Rechtecken mit abgerundeten Ecken, wobei alle Umriss- und Fülleigenschaften des normalen Rechtecks zur Verfügung stehen
- **Polygon**: Zeichnen von Polygonen mit verschiedenen Füllfarben/-mustern sowie Linienstärken/-arten für den Umriss des Objektes
- **Text**: Eingabe von Text in verschiedenen Schriftgrößen/-farben

- **Radierer**: Überdecken ausgewählter Teile einer Zeichnung durch die Hintergrundfarbe des Dokumentes
- **Farbauswahl anhand existierender Objekte**: Zuweisung von Stift- und Füllfarbe mittels Selektion dieser anhand bereits existierender Objekte innerhalb des Dokumentes (Funktionalität wird in den meisten Bildbearbeitungsprogrammen durch den Begriff *Pipette* beschrieben)
- **Farbfüller**: Zuweisung einer Füllfarbe und eines Füllmusters für Kreise/Ellipsen, Rechtecke und Polygone.

Alle hier aufgeführten Funktionalitäten lassen sich in ihrer Ausführung durch die Angabe der nachfolgend aufgeführten Informationen beschreiben:

- **Zeichenoperation**: bezeichnet eine der zum Funktionsumfang gehörigen Zeichenoperation, die der Anwender ausführen kann
- **Startkoordinate**: bezeichnet den Startpunkt für eine Linie oder einzufügenden Text beziehungsweise den Ausgangspunkt für ein umgebendes Rechteck, welches ein Objekt in Form eines Kreises, einer Ellipse oder eines Rechtecks beschreibt
- **Endkoordinate**: bezeichnet den Endpunkt für eine Linie beziehungsweise den Endpunkt für ein umgebendes Rechteck, welches ein Objekt in Form eines Kreises, einer Ellipse oder eines Rechtecks beschreibt
- **Linienfarbe**: bestimmt die Farbe einer Linie beziehungsweise des Umrisses eines Objektes
- **Linienart**: bestimmt den Linientyp (z.B. durchgezogene Line, Strich-Punkt-Linie, gestrichelte Linie, ...) für Linien und Objektumrisse
- **Linienstärke**: bestimmt die Linienbreite für Linien und Objektumrisse
- **Füllfarbe**: bestimmt die Füllfarbe eines Objektes
- **Füllmuster**: bestimmt das Füllmuster eines Objektes
- **Schriftfarbe**: bestimmt die Schriftfarbe eines eingegebenen Textes
- **Schriftgröße**: bestimmt die Schriftgröße eines eingegebenen Textes.

Wie aus dieser Auflistung ersichtlich wird, sind also lediglich die Art der Operation, zwei Punktkoordinaten und die Format-Optionen für Linie/Umriss, Füllung und Schrift erforderlich, um jede der im vorliegenden Konzept verfügbaren Zeichenoperationen beschreiben zu können. Eine Ausnahme bildet hier die Funktion zum Einfügen von Text. Hier ist neben den

bereits angegebenen Informationen außerdem noch der betreffende Text selbst als Information erforderlich.

Eine kompakte Beschreibung der vom Anwender ausgeführten Operationen ist bezüglich des damit verbundenen Datenaufkommens von außerordentlicher Bedeutung, da der Whiteboard-Dienst analog zum in *Kapitel 3.3.2* beschriebenen Chat-Dienst auf einer Client/Server-Architektur aufbaut und somit vom Whiteboard-Server sämtliche Zeichenoperationen eines Whiteboard-Clients an alle zu dessen Gruppe gehörigen Mitglieder verteilt werden müssen. Zur Berechnung des dabei zu erwartenden Datenaufkommens am Server werden folgende Festlegungen getroffen, welche die bei den einzelnen Operationen verfügbaren Eigenschaftswerte begrenzen und damit die Größe der damit verbundenen Information bestimmen:

- Die maximal verfügbare Zeichenfläche eines Dokumentes ist auf die Größe von 1000 * 1000 Pixel festgelegt. Zur Kodierung einer **Start- bzw. Endkoordinate**, bestehend aus je einem Wert für die X-Achse und Y-Achse, sind somit **20 bit** erforderlich.

- Der **Funktionsumfang der Zeichenoperationen** umfasst 9 Funktionen. Somit werden **4 bit** zur Kodierung benötigt.

- Es stehen 5 verschiedene **Linienarten** zur Verfügung. Zur Kodierung werden hierfür **3 bit** benötigt.

- Die zur Verfügung stehenden **Linienstärken** für Linien und Umrisse bewegen sich im ganzzahligen Bereich von 1 bis 16 Punkten. Somit werden **4 bit** zur Kodierung verwendet.

- Die Anzahl der zum Zeichnen verfügbaren **Linien-, Füll- und Schriftfarben** ist auf jeweils 16 begrenzt. Deren RGB-Werte sind in einer dem Client und dem Server zur Verfügung stehenden Farbtabelle definiert. Zur Kodierung sind also **4 bit** erforderlich.

- Die zur Verfügung stehende **Anzahl von Füllmustern** für Objekte beträgt 8. Es werden **3 bit** zur Kodierung benötigt.

- Die für das Einfügen von Text wählbare **Schriftgröße** bewegt sich im Bereich von 8 bis 48. Somit werden hierfür **6 bit** zur Kodierung verwendet.

Da beim Zeichnen einer Linie oder eines anderen Objektes keine Angaben für Schriftfarbe und -größe benötigt werden und im Falle des Einfügens von Text keine Linienfarbe und -stärke, werden die entsprechenden Speicherstellen in der zu übermittelnden Datenstruktur dementsprechend doppelt belegt. In *Abbildung 3.8*, welche die für den Informationstransport benötigte Datenstruktur darstellt, wird dies deutlich. Außerdem kann man hier auch erkennen, dass zur Übermittlung einer Zeichenoperation Daten im Gesamtumfang von 64 bit anfallen. Die einzige

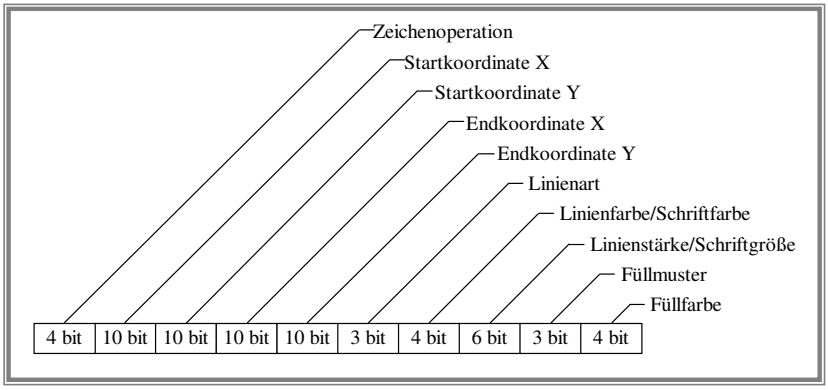

Abbildung 3.8: Datenstruktur zur Übermittlung der Whiteboard-Daten

Ausnahme bildet hierbei die Operation zum Einfügen von Text, bei welcher zusätzlich noch die Zeichenfolge des betreffenden Textes übertragen wird.

Anhand dieser Aussagen wird die zu erwartende Netzlast am Server an einem Beispiel berechnet. Analog zur Berechnung dieser Last bei dem im Rahmen dieses Projektes entworfenen Chat-Dienstes wird auch hier zu Grunde gelegt, dass zur Datenverteilung Unicast verwendet wird, ein Nutzer maximal 2 Zeichenoperationen pro Sekunde ausführen kann und 100 Benutzer am Whiteboard-Dienst angemeldet sind. Somit ergibt sich eine Datenmenge von maximal 128 bit/s, die jeder Whiteboard-Teilnehmer an den Whiteboard-Server versendet. Sind die 100 Nutzer beispielsweise gleichmäßig auf 5 Gruppen verteilt, so hat der Server für die von einem einzelnen Teilnehmer ausgeführten Zeichenoperationen eine Datenmenge von insgesamt *2 Operationen/s * 20 Teilnehmer * 64 bit/Operation = 2,56 Kbit/s* an die zugehörige Gruppe zu verteilen. Führen hingegen alle Mitglieder gleichzeitig die maximale Anzahl von Operationen pro Sekunde aus, so erhöht sich das Datenaufkommen beim Server auf *2,56 Kbit/s * 20 = 51,2 Kbit/s* pro Gruppe. Für alle 5 Gruppen sind es also *51,2 Kbit/s *5 = 256 Kbit/s*, die der Whiteboard-Server zu verteilen hat. Die nachfolgende *Tabelle 3.3* zeigt die Ergebnisse der Berechnungen für die Aufteilung der 100 Nutzer auf 1 beziehungsweise 5 Gruppen.

In der Praxis ist nicht zu erwarten, dass, wie im Beispiel aufgeführt, 20 oder gar 100 Anwender gleichzeitig Zeichenoperationen innerhalb einer Gruppe ausführen, denn dies würde kein effektives Arbeiten zulassen. Auch ist die angenommene Größe einer Gruppe mit 100 Nutzern hoch gegriffen und praxisfremd. Die Berechnungen zeigen aber, dass das im Beispiel anfallende Datenaufkommen in einer Größenordnung von 1.28 Mbit/s beim Server trotzdem noch beherrschbar ist.

Abweichend von den hier aufgeführten Berechnungen liegt das tatsächliche Datenaufkommen über dem hier berechneten, da vom TCP- und vom IP-Protokoll zur Datenübermittlung neben den Nutzdaten, also im vorliegenden Fall den Whiteboard-Informationen, weitere Header-Daten in Form von Protokollinformationen übermittelt werden. Diese Header-Daten werden

Angaben	**100 Benutzer, 1 Gruppe**	**100 Benutzer, 5 Gruppen**
Zeichengeschwindigkeit eines Benutzers	2 Operationen/s	2 Operationen/s
Anzahl der Benutzer in einer Gruppe	100	20
Belastung des Netzwerks durch 1 Benutzer	12,8 Kbit/s	2.56 Kbit/s
Datendurchsatz pro Gruppe, maximal	1.28 Mbit/s	51.2 Kbit/s
Gesamte Belastung des Netzwerks, maximal	1.28 Mbit/s	256 Kbit/s

Tabelle 3.3: Datendurchsatz eines Whiteboard-Server-Moduls (ohne Textdaten)

in die Berechnungen aber nicht mit einbezogen, da ihre Größe von verschiedenen Parametern abhängig ist.

Auf die Funktionalität der in Bildbearbeitungsprogrammen häufig vorkommenden *Freihandlinie* wird an dieser Stelle bewusst verzichtet. Grund hierfür ist die Tatsache, dass dabei jede bei dieser Operation entstehende „Teillinie“ durch ihre Punktkoordinaten beschrieben werden muss, was bei entsprechend umfangreichen Freihandlinien zu einem erheblichen Datenaufkommen führt.

Der Whiteboard-Server hat neben der hier beschriebenen Verteilung von Zeichenoperationen noch eine weitere Aufgabe zu erfüllen. Tritt ein Nutzer einer bereits bestehenden Gruppe bei, so muss ihm der Server als Erstes den aktuellen Arbeitsstand der Gruppe, also das Dokument in der augenblicklichen Beschaffenheit, übermitteln. Hierzu gibt es zwei Möglichkeiten:

- Übermittlung des Dokumentes in Form einer komprimierten Grafik, beispielsweise im JPEG-Format
- Übermittlung aller bisher von der Gruppe ausgeführten Zeichenoperationen, damit der neu hinzugekommene Client diese nachträglich ausführen kann.

Unabhängig davon, welche der beiden Möglichkeiten Anwendung findet, muss der Server die bei ihm eintreffenden Zeichenoperationen nach Gruppen getrennt lokal nachvollziehen beziehungsweise in Listenform abspeichern. Betrachtet man die Arbeitsfläche des Dokumentes mit einer Größe von 1000 * 1000 Pixeln, so stellt man fest, dass selbst bei einer geringen Anzahl von Zeichenoperationen eine komprimierte Grafik vom Datenumfang her wesentlich größer ist, als die Gesamtzahl der im System verwendeten Datenstrukturen, welche das Dokument beschreiben. Eine solche Datenstruktur umfasst lediglich 64 bit. Die Übermittlung des

Arbeitsstandes in Form einer komprimierten Grafik kommt hier also nicht in Betracht, da das Datenaufkommen ungleich höher ist und die nachträgliche Ausführung von Zeichenoperationen auf Client-Seite auch nur minimal Rechenzeit erfordert.

Zum Zweck der Sammlung von Zeichenoperationen durch den Server ordnet dieser jeder Gruppe zwei Listen zu. Während in der ersten Liste die Datenstrukturen einer Gruppe in der Reihenfolge des Eintreffens beim Server abgelegt werden, enthält die zweite Liste die empfangenen Zeichenketten, welche beim Einfügen von Text in das Dokument anfallen. Beim Senden aller Datenstrukturen an einen Nutzer, der einer bestehenden Gruppe betritt, überprüft der Server zunächst die Datenstruktur auf die darin vermerkte Zeichenoperation, bevor er sie sendet. Bezeichnet diese das Einfügen von Text, so übermittelt der Server unmittelbar auf diesen Sendevorgang die in der zweiten Liste abgespeicherte Zeichenkette, bevor die restlichen Datenstrukturen an den Client versendet werden. Die Reihenfolge der Zeichenketten in der zweiten Liste stimmt mit der Reihenfolge ihrer zugehörigen Datenstrukturen aus der ersten Liste überein, so dass eine korrekte Zuordnung sichergestellt wird.

3.3.5 FTP und eMail

Mit den Möglichkeiten des Datentransfers per FTP und des Informationsaustausches via eMail werden dem Nutzer zwei weitere Interaktionskomponenten zur Verfügung gestellt. Der hierbei verwendete FTP-Dienst ermöglicht es dem Nutzer, auf einfache Art und Weise Dateien von einem entsprechenden FTP-Server herunterzuladen, welche beispielsweise begleitend zum Vorlesungsinhalt angeboten werden. Da dieser Dienst keine Interaktion mit anderen im System befindlichen Nutzern beinhaltet, muss seine Ausführung auch nicht vom Sitzungsmanager des zentralen Servers verwaltet werden, so dass hierfür auch keine Anbindung an den zentralen Server notwendig ist. Verwendet wird für diesen Zweck ein eigenständig lauffähiger FTP-Client, welcher im GWOTS lediglich aus der Oberfläche des zentralen Clients heraus gestartet wird. Alle weiteren Eingaben und Aktionen am FTP-Client erfolgen danach unabhängig vom zentralen Client.

Die gleiche Vorgehensweise zum Starten des Dienstes gilt auch für den vom Distance-Learning-System angebotenen eMail-Dienst, nur das hier anstatt des FTP-Clients der auf dem aktuell verwendeten Computer registrierte Standard-Mail-Client vom zentralen Client aus aufgerufen wird.

3.3.6 Kurznachrichten

Der Dienst zum Austausch von Kurznachrichten ermöglicht es dem Anwender, kurze Mitteilungen an einen anderen am System angemeldeten Nutzer zu verschicken. Der Sinn dieses Dienstes besteht dabei darin, dass sich Nutzer untereinander auf einfache und schnelle Art und Weise verständigen können, um beispielsweise den Vorschlag zur Eröffnung einer neuen Grup-

pe im Whiteboard-Dienst zu unterbreiten oder einen anderen Nutzer zum Beitritt in eine bereits bestehende Gruppe aufzufordern.

Eine Kurznachricht kann an einen einzelnen Nutzer versendet werden, jedoch nicht an eine Gruppe, wie das etwa beim Chat möglich ist. Hierzu wählt der Anwender zunächst aus einer ihm zur Verfügung stehenden Liste, welche alle am System angemeldeten Nutzerkennungen enthält, den Empfänger aus und schreibt seine Mitteilung in ein hierfür vorgesehenes Eingabefeld, bevor das Senden der Nachricht vollzogen wird.

In der grafische Oberfläche des zentralen Client ist ein Nachrichtenfenster integriert, welches die Mitteilungen permanent angezeigt. Somit bleiben empfangene Kurznachrichten für den Empfänger während der gesamten Sitzung erhalten. Dieses Anzeigefenster dient gleichzeitig auch zur Anzeige von Systemmeldungen, die vom zentralen Server und Client versendet werden, wobei hier eine optische Trennung der Systemmeldungen von den Kurznachrichten durch die Verwendung unterschiedlicher Schriftfarben stattfindet. Der Empfang einer Nachricht wird dem Empfänger durch ein akustisches Signal mitgeteilt.

Der Kurznachrichten-Dienst ist fester Bestandteil des zentralen Client. Die Mitteilungen werden hier genau wie die von den Steuernachrichten übermittelten Systemmeldungen über den Haupt-Socket zwischen zentralem Client und Server übertragen. Ein zusätzlicher Dienst-Socket ist hierfür nicht vorhanden, da dieser Dienst lediglich Nachrichten in begrenzter Größe versenden kann. Damit ist das dabei zu erwartende Datenaufkommen als gering einzustufen.

3.4 Konzeptvergleich zum System OVID

Analog zum OVID werden auch im GWOTS neben dem Dienst zur Präsentation der Vorlesungsinhalte die Zusatzdienste Chat, Whiteboard, Videokonferenz, eMail und FTP angeboten. Dabei wird zur Verwaltung dieser Dienste ebenfalls ein einheitliches Sitzungsmanagement verwendet, wobei die hierzu erforderliche Kommunikationsstruktur allerdings vereinfacht wird (vgl. *Abbildung 2.3* und *Abbildung 3.3*). Erreicht wird dies dadurch, dass im Gegensatz zum OVID, bei dem die Dienst-Client-Module und Dienst-Server-Module der Zusatzdienste als eigenständig lauffähige Module in das System eingebunden werden, die Dienst-Client-Module für Chat, Whiteboard und Videokonferenz beim GWOTS fest in den zentralen Client integriert werden. Gleiches gilt im GWOTS auch für die Dienst-Server-Module des Chat- und Whiteboard-Dienstes in Bezug auf den zentralen Server. Lediglich der Videokonferenz-Server bleibt vom zentralen Server unabhängig (vgl. *Abbildung 3.3*) und wird bei seiner Nutzung als eigenständig lauffähiges Programm ausgeführt, um eine Verwendung dieses und des zentralen Servers auf getrennten Rechnersystemen zu ermöglichen. Das hat den Vorteil, dass beim Einsatz beider Server auf getrennten Rechnersystemen mit separater Netzanbindung bei entsprechend starker Frequentierung des Videokonferenz-Servers die Netzanbindung des zentralen Servers hiervon nicht belastet wird. Die Integration der angesprochenen Dienst-Server-Module

in den zentralen Server gestattet dem GWOTS die gemeinsame Verwendung des im zentralen Server enthaltenen Sitzungsmanagers, wohingegen diese Dienst-Server-Module des OVID jeweils einen eigenen Sitzungsmanager besitzen und deshalb vom jeweiligen Dienst-Client-Modul aus ein zusätzlicher Kommunikationskanal zum zentralen Server aufgebaut werden muss (vgl. *Abbildung 2.3*), welcher im OVID das einheitliche Sitzungsmanagement ermöglicht. Im GWOTS ist dieser zusätzliche Kommunikationskanal nicht erforderlich, da das Sitzungsmanagement für alle hier beschriebenen Dienste vom gemeinsam genutzten Sitzungsmanager des zentralen Servers erledigt wird. Die Integration der Dienst-Client-Module in den zentralen Client gestattet im GWOTS außerdem die gemeinsame Nutzung der Socket-Verbindung zwischen zentralem Client und zentralem Server zum Austausch von Steuernachrichten (vgl. *Abbildung 3.3*).

Neben der vereinfachten Kommunikationsstruktur zum einheitlichen Sitzungsmanagement bringt die Konzeption des GWOTS auch den Vorteil mit sich, dass die grafischen Benutzerschnittstellen der Zusatzdienste aufgrund der beschriebenen Integration in die Benutzeroberfläche des zentralen Clients eingebunden werden können und somit ein einheitliches und kompaktes Erscheinungsbild der Client-Anwendung gegeben ist. Im OVID ist dies aufgrund der Eigenständigkeit dieser Zusatzdienstmodule nicht möglich.

Bei der Abmeldung eines Nutzers vom Distance-Learning-System ist beim OVID der zentrale Server dafür verantwortlich, dass eventuell noch aktive Zusatzdienste dieses Nutzers beendet werden. Hierzu wird über den zusätzlichen Kommunikationskanal zu dem noch aktiven Dienst-Client-Modul vom zentralen Server aus eine Steuernachricht versendet (vgl. *Abbildung 2.3*), welche das Dienst-Client-Modul zur Beendigung auffordert. Im GWOTS hingegen erübrigt sich diese zusätzliche Kommunikation zwischen dem zentralen Server und den Dienst-Client-Modulen, da die Dienst-Client-Module im zentralen Client integriert sind und bei einer Abmeldung vom System automatisch vom zentralen Client aus beendet werden.

Die im *Kapitel 3.1.2* vorgestellte Struktur der im GWOTS verwendeten Datenbank unterscheidet sich von der des OVID (vgl. *Tabelle 2.1*) dadurch, dass einerseits Attribute hinzugekommen sind und andererseits eine Aufteilung der Attribute auf insgesamt drei Relationen erfolgt, während im OVID nur eine einzige Relation zur Aufnahme der Daten verwendet wird. Die Ausgliederung der Attribute für den Fachbereichsnamen beziehungsweise den Matrikelnamen in separate Relationen bringt gegenüber der Datenbank des OVID den Vorteil mit sich, dass die Zeichenketten, welche die Matrikel- und Fachbereichsnamen enthalten, nur ein einziges Mal in der Datenbank vermerkt werden müssen und die Zuordnung dieser Daten zu den jeweiligen Nutzern über die im *Kapitel 3.1.2* beschriebene Fremdschlüsselbeziehung erfolgt. Hierdurch wird Speicherplatz in der Datenbank eingespart, da in der Regel mehrere Benutzer zu einem Matrikel sowie mehrere Matrikel zu einem Fachbereich gehören und der Fremdschlüssel weniger Speicherplatz benötigt als die Zeichenkette, auf die er verweist. Außerdem wird der Fachbereich im hier entworfenen Konzept der Relation *Matrikel* zugeordnet

und muss somit nicht jedem einzelnen Nutzer-Datensatz zugeordnet werden, wie dies bei der Datenbank des OVID der Fall ist. Ein weiterer Vorteil des Datenbankkonzeptes im GWOTS besteht auch darin, dass bei einer Modifikation eines Matrikel- oder Fachbereichsnamen nur die betreffende Zeichenkette des einzelnen hierzu gehörigen Attributes in der Relation *Matrikel* beziehungsweise *Fachbereich* geändert wird, während beim OVID in solch einem Fall alle Nutzerdatensätze modifiziert werden müssen, die dem betroffenen Matrikel beziehungsweise Fachbereich angehören.

Eines der hinzugekommen Attribute ist das in der Relation *Matrikel* enthaltene Feld *Start-URL*, welches die Zuordnung einer Startseite in Abhängigkeit vom Matrikel realisiert. Während beim OVID die Startseite zum Präsentationsdienst die Verweise für die Vorlesungsinhalte sämtlicher Fachbereiche und Matrikel enthält, wird dem Anwender im GWOTS durch matrikelbezogene Startseiten die Auswahl der für ihn vorbereiteten Vorlesungsinhalte erleichtert.

Auch die in der Relation *Nutzer* vorhandenen Attribute *Basis*, *Chat* und *Whiteboard* sind gegenüber der im OVID verwendeten Datenbank hinzugekommen. Diese dienen zur dauerhaften Speicherung für die Berechtigung eines Anwenders zur Nutzung des Basisdienstes sowie der Zusatzdienste Chat und Whiteboard. Im OVID kann ein Anwender zwar zur Beendigung der Nutzung eines dieser Dienste aufgefordert werden, jedoch ist dort keine dauerhafte Sperrung möglich.

Zur Verwaltung der in der Datenbank abgespeicherten Nutzer-, Matrikel- und Fachbereichsdaten ist im zentralen Server des GWOTS ein grafisches Benutzerinterface vorhanden, welches sowohl die Einsicht dieser Daten als auch deren Manipulation auf einfache Weise erlaubt. Diese grafische Schnittstelle ist beim OVID nicht vorhanden, aber zu einer komfortablen Verwaltung der in der Datenbank enthaltenen Daten durch den Administrator unabdingbar.

Ein gravierender Unterschied besteht gegenüber dem System OVID in der Unterstützung schmalbandiger Netzanbindungen für den Dienst zur Präsentation von Vorlesungsinhalten. Während dieser Dienst bei der Integration von Audio/Video-Daten in die Vorlesunginhalte im OVID nur bei Verwendung einer Netzanbindung mit einer Bandbreite von mindestens 1.2 Mbit/s [ROS00] genutzt werden kann, reicht beim System GWOTS auch eine Modemverbindung mit einer Bandbreite von 56 Kbit/s aus. Somit wird dieser Dienst nach dem derzeitigen Stand der Netzanbindungen im Heimbereich durch das GWOTS praxistauglich gemacht, was beim OVID infolge der dort erforderlichen Bandbreite nicht der Fall ist. Erreicht wird dies durch die Verwendung eines Streaming-Servers, welcher die angesprochenen Audio/Video-Daten in Form von Multi-Stream-Dateien in verschiedenen Qualitätsstufen mit unterschiedlichen Gesamt-Bitraten bereitstellt und dem anfordernden Client entsprechend dessen verfügbarer Bandbreite übermittelt. Der im OVID zur Bereitstellung der Ausio/Video-Dateien verantwortliche Web-Server kann diese Funktionalität der bandbreitenabhängigen Übertragung nicht realisieren.

Während beim OVID für den Videokonferenz-Dienst die Untersuchung der Möglichkeit

zur Übertragung von Audio/Video-Daten auf herkömmlichen PC-Systemen bei höchstmöglicher Qualität im Vordergrund steht und dabei auch spezielle Hardware in Form von Videoschnittkarten verwendet wird, zielt das Konzept dieses Dienstes im GWOTS darauf ab, den Videokonferenz-Dienst auch für schmalbandige Netzanbindungen und ohne spezielle Hardware verfügbar zu machen. Beim OVID wird zur Nutzung dieses Dienstes seitens des Clients eine Bandbreite von mindestens 1.2 Mbit/s [ROS00] benötigt, was eine Nutzung im Heimanwenderbereich jedoch verhindert. Deswegen wird im GWOTS das H.323 Standardprotokoll zur Übertragung der Audio/Video-Daten verwendet. Der Vorteil dieses Konzeptes besteht darin, dass somit auch eine Nutzung des Dienstes bei Verwendung einer schmalbandigen Modemanbindung ermöglicht wird. Als Nachteil ist dabei aber gegenüber der im OVID verwendeten Technologie eine geringere Bildqualität in Kauf zu nehmen.

Eine Verbesserung bezüglich des im OVID integrierten Videokonferenz-Dienstes stellt im GWOTS auch die Möglichkeit zur Gruppenkommunikation dar, während im OVID lediglich zwei Gesprächspartner über diesen Dienst miteinander kommunizieren können. Im GWOTS ist beides möglich, wobei im Gegensatz zu einer privaten Konferenz, bei welcher der Datenaustausch direkt zwischen den Videokonferenz-Komponenten der beiden Teilnehemer erfolgt, in einer Gruppenkonferenz der Informationsaustausch über einen zusätzlichen Videokonferenz-Server (MCU) abläuft.

Die Übertragung von Listen zwischen dem zentralen Server und dem zentralen Client, welche die verfügbaren Gruppennamen der einzelnen Zusatzdienste enthalten, erfolgt analog zu dieser Funktionalität des OVID. Diese Listen sind notwendig, um dem Anwender den Betritt in bereits bestehende Zusatzdienstgruppen zu ermöglichen. Im GWOTS werden gegenüber dem OVID aber zusätzlich zu diesen hier beschriebenen Listen noch hiermit korrespondierende Nutzerlisten übertragen, welche die Nutzkennungen der zu den jeweiligen Gruppen gehörigen Mitgliedern enthalten. Der Vorteil hierbei besteht darin, dass dem Anwender damit die Möglichkeit gegeben wird, sich einen Überblick darüber zu verschaffen, wer in welcher Gruppe aktiv ist. Diese Funktionalität erhöht sicherlich auch die notwendige Akzeptanz beim Anwender.

Um das Datenaufkommen beim Chat-Dienst in Hinsicht auf die beim Chat-Server anfallende Netzlast so gering wie möglich zu halten, wird der zum Austausch von Chat-Nachrichten erforderliche Header im GWOTS bezüglich seiner Größe optimiert. Während dieser Header im OVID 104 bit umfasst und dabei die Nutzerkennung des Verfassers der Nachricht sowie die Angabe der darzustellenden Schriftfarbe enthält, beträgt die Größe dieses Headers im GWOTS aufgrund des hierzu beschriebenen Konzeptes lediglich 8 bit. Der Vorteil liegt dabei klar in der geringeren Datenmenge einer Chat-Nachricht, was sich positiv auf die beim Chat-Server auftretenden Netzlast auswirkt, welche beim Verteilen der Daten entsteht.

Kapitel 4

Implementierung der Konzeptionen

Im Rahmen dieses Kapitels wird die programmiertechnische Umsetzung der in dieser Arbeit vorgestellten Konzeptionen behandelt. Zu diesem Zweck werden die für dieses System erstellten und verwendeten Klassen ausführlich beschrieben sowie dabei angewandte Methoden für entsprechende Schlüsselstellen an Beispielen erklärt. Diese Erläuterungen erfolgen für den Server-seitigen Einsatz und die Verwendung auf Seite des Clients aus Gründen der Übersichtlichkeit so weit als möglich getrennt und werden dementsprechend gekennzeichnet.

Als Programmierumgebung wird Borland Delphi™in der Version 6 verwendet, welche die Programmiersprache *Object Pascal* beinhaltet. Die Wahl hierfür begründet sich aus der Tatsache, dass die darin integrierte VCL-Klassenbibliothek eine umfangreiche Sammlung von Objekten zur Kapselung der Windows-API sowie weitere Programmierunterstützung enthält, und sich somit auf einfache Weise 32-Bit-Anwendungen für Windows bei minimalem Programmieraufwand erstellen lassen. Die visuelle Programmierumgebung ist hierbei durchweg objektorientiert [BOR01]. Außerdem sind eine Vielzahl weiterer Komponenten frei verfügbar, welche von unabhängigen Entwicklern erstellt wurden und nahezu jeden Anwendungsbereich abdecken.

Zur netzwerkseitigen Übertragung von Daten werden im vorliegenden System TCP-Sockets verwendet, da einerseits laut Konzeption eine Punkt-zu-Punkt-Verbindung zwischen zentralem Client und Server über die gesamte Sitzungsdauer hinweg gehalten werden muss *(vgl. Kapitel 3.1, Seite 21)*, und zum anderen die hierbei übermittelten Daten vollständig sein müssen. Das dabei verwendete TCP-Protokoll sichert die erforderliche Vollständigkeit und Fehlerfreiheit der Übertragungsdaten ab, indem fehlende oder fehlerhafte Datenpakete von der Gegenseite erneut angefordert werden.

Bevor die Implementierungen für das Sitzungsmanagement, die Ressourcenverwaltung und die einzelnen Interaktionskomponenten vorgestellt werden, folgen im nächsten Kapitel zunächst einige Ausführungen zur Programmierung der bereits angesprochenen Sockets innerhalb der hier verwendeten Programmierumgebung, um das nötige Verständnis für die damit verbunde-

nen Routinen zu vermitteln, welche innerhalb der Implementierungen für die einzelnen Interaktionskomponenten immer wieder vorkommen.

4.1 Socket-Programmierung unter Borland Delphi 6

Zum Aufbau und zur Verwaltung von TCP/IP-Socket-Verbindungen zwischen Netzwerkanwendungen stehen unter Borland Delphi™ unter anderem die beiden im vorliegenden System verwendeten Komponenten *TServerSocket* und *TClientSocket* zur Verfügung. Jede dieser Komponenten ist mit Socket-Objekten verbunden, welche den Endpunkt einer Socket-Verbindung darstellen und zur Datenübertragung das TCP/IP-Protokoll verwenden. Diese Objekte werden hierbei eingesetzt, um Server-Socket-Anforderungen zu kapseln. Während über die zum Socket-Objekt gehörigen Eigenschaften Einstellungsdetails, wie beispielsweise die IP-Adresse der Server-Anwendung und die zugehörige Port-Nummer, festgelegt werden können, dienen die mit dem Socket-Objekt verbundenen Ereignisse zur Auswertung und Verarbeitung von Übertragungsdaten beziehungsweise dabei aufgetretenen Fehlern [BOR01]. Die wichtigsten Eigenschaften, Methoden und Ereignisse werden nachfolgend getrennt für Client und Server vorgestellt.

Client-Sockets

Durch das Hinzufügen eines TClientSocket-Objektes zu einer Anwendung wird diese zu einem TCP/IP-Client. Um über dieses Objekt den Verbindungsaufbau und Datenaustausch zu beziehungsweise mit einem TCP/IP-Server zu ermöglichen, muss es zunächst entsprechend initialisiert werden. Hierzu stellt das Objekt die folgenden Eigenschaften zur Verfügung [BOR01]:

- **Address**: Hierüber wird die IP-Adresse des Server-Systems bestimmt, zu dem eine Verbindung aufgebaut werden soll.
- **Host**: Diese Eigenschaft dient zur Festlegung der URL, über die das Server-System erreichbar ist. Die Angabe dieser URL ist optional und hat gegenüber der Eigenschaft *Address* Vorrang.
- **Port**: Hierüber wird die Port-Nummer festgelegt, über welche das Server-System auf dem Zielrechner angesprochen wird.
- **ClientType**: Diese Eigenschaft dient zur Bestimmung der Art der Verbindung. Diese bestimmt, ob der Client-Windows-Socket Informationen über die Socket-Verbindung synchron oder asynchron liest und schreibt.
- **Socket**: Diese Eigenschaft beschreibt den Client-Endpunkt der Windows-Client-Socket-Verbindung und dient somit zum Lesen und Schreiben von Informationen über die Socket-Verbindung mittels der von diesem Objekt bereitgestellten Methoden.

Zum Auf- beziehungsweise Abbau einer Verbindung von der Client-Anwendung zum Server-System bietet das TClientSocket-Objekt die Methoden *Open* beziehungsweise *Close* an. Beim Aufruf von *Open* wird dabei die Socket-Verbindung zunächst anhand der zuvor beschriebenen Eigenschaften initialisiert, bevor nach dem Server-System gesucht wird. Verläuft diese Suche erfolgreich, so wird die Verbindung zum Server hergestellt und die Client-Anwendung ist für Lese- und Schreiboperationen bereit.

Um auf einen erfolreichen beziehungsweise gescheiterten Verbindungsaufbau, eingehende beziehungsweise ausgehende Daten, eventuell auftretende Fehler und den Verbindungsabbau reagieren zu können, löst das TClientSocket-Objekt entsprechend der Situation Ereignisse aus. Diese Ereignisse werden nachfolgend aufgezählt und näher beschrieben [BOR01]:

- **OnConnect(Sender: TObject; Socket: TCustomWinSocket)**: Dieses Ereignis wird ausgelöst, wenn eine Verbindung zum Server geöffnet wird. Innerhalb der zugehörigen Behandlungsroutine können Aktionen festgelegt werden, die unmittelbar nach einem erfolgreichen Verbindungsaufbau ausgeführt werden sollen. Der Parameter *Socket* bietet hierbei - wie übrigens auch bei allen anderen Ereignisroutinen des TClientSocket-Objektes - Zugriff auf das Socket-Objekt selbst, also den Endpunkt dieser Verbindung innerhalb der Client-Anwendung.

- **OnConnecting(Sender: TObject; Socket: TCustomWinSocket)**: Das Auslösen dieses Ereignisses durch das TClientSocket-Objekt erfolgt, nachdem der Server lokalisiert wurde, aber noch vor dem eigentlichen Verbindungsaufbau. Aktionen, welche direkt vor dem Verbindungsaufbau ausgelöst werden müssen, sind in der zugehörigen Behandlungsroutine dieses Ereignisses festzulegen.

- **OnDisconnect(Sender: TObject; Socket: TCustomWinSocket)**: Ein Auslösen dieses Ereignisses erfolgt unmittelbar vor dem Schließen einer Verbindung zu einem Server-Socket durch den Client-Socket. In der zugehörigen Behandlungsroutine können also Aktionen definiert werden, die kurz vor dem Verbindungsabbau zu einem Server-Socket ausgeführt werden müssen.

- **OnError(Sender: TObject; Socket: TCustomWinSocket; ErrorEvent: TErrorEvent; var ErrorCode: Integer)**: Dieses Ereignis wird ausgelöst, wenn der Socket eine Verbindung nicht herstellen, benutzen oder beenden kann. Somit kann bei Bedarf innerhalb der Behandlungsroutine auf den im Parameter *ErrorEvent* definierten Fehler reagiert werden.

- **OnLookup(Sender: TObject; Socket: TCustomWinSocket)**: Ein Auslösen dieses Ereignisses erfolgt, wenn ein Client-Socket nach einem Server-Socket sucht, zu dem die Verbindung hergestellt werden soll. Dementsprechend können in der zugehörigen Be-

handlungsroutine Aktionen festgelegt werden, die eventuell vor einer solchen Suche ausgelöst werden müssen.

- **OnRead(Sender: TObject; Socket: TCustomWinSocket)**: Dieses Ereignis tritt auf, wenn ein Client-Socket Daten aus der Socket-Verbindung lesen soll. Innerhalb der zugehörigen Behandlungsroutine müssen also die vom Parameter *Socket* bereitgestellten Methoden zum Lesen der vorhandenen Daten aufgerufen werden.
- **OnWrite(Sender: TObject; Socket: TCustomWinSocket)**: Ein Auslösen dieses Ereignisses erfolgt, wenn ein Client-Socket Daten in die Socket-Verbindung schreibt.

Um Daten aus dem beziehungsweise in den Socket zu lesen beziehungsweise zu schreiben, stellt das bereits beschriebene Socket-Objekt die folgenden Methoden zur Verfügung [BOR01]:

- **ReceiveText**: Diese Methode liest eine Zeichenkette aus der Socket-Verbindung aus.
- **ReceiveLength**: Die Anzahl von Bytes, die über die Socket-Verbindung gesendet werden, wird mittels dieser Methode ermittelt.
- **ReceiveBuf(var Buf;Count:Integer)**: Mittels dieser Methode werden *Count* Bytes aus der Socket-Verbindung ausgelesen und im Parameter *Buf* abgelegt. Die Anzahl der Bytes, die aus der Verbindung gelesen werden sollen, wird mit der Methode ReceiveLength festgelegt.
- **SendText(const S: String)**: Diese Methode schreibt eine in *S* enthältene Zeichenkette in die Socket-Verbindung.
- **SendBuf(var Buf; Count: Integer)**: Das Schreiben von *Count* Bytes aus dem Parameter *Buf* in die Socket-Verbindung erfolgt mittels dieser Methode.
- **SendStream(AStream: TStream)**: Diese Methode schreibt alle Informationen, die aus dem im Parameter *AStream* angegebenen Stream gelesen werden können, in die Socket-Verbindung.

Server-Sockets

Das TServerSocket-Objekt muss analog zum TClientSocket-Objekt zunächst mit den entsprechenden Eigenschaftswerten initialisiert werden, bevor die Anforderungen einer TCP/IP-Verbindung von anderen Maschinen entgegengenommen werden können. Die wichtigsten Eigenschaften werden nachfolgend aufgezählt und kurz erläutert [BOR01]:

- **Port**: Hierüber wird die Port-Nummer festgelegt, über welche das Server-System auf dem Zielrechner erreichbar ist.

- **ServerType**: Diese Eigenschaft dient zur Bestimmung der Art der Verbindung. Diese bestimmt, ob der Server-Socket Informationen über die Socket-Verbindung in asynchroner oder synchroner Form liest und schreibt.

- **Socket**: Diese Eigenschaft beschreibt den empfangenden Server-Endpunkt der Windows-Server-Socket-Verbindung und dient somit zum Lesen und Schreiben von Informationen über die Socket-Verbindung mittels der von diesem Objekt bereitgestellten Methoden. Außerdem werden Eigenschaften, Methoden und Ereignisse bereitgestellt, welche die Verwaltung mehrerer Socket-Verbindungen zu verschiedenen Client-Sockets gestatten.

Zum Auf- beziehungsweise Abbau einer Verbindung bietet das TServerSocket-Objekt analog zum TClientSocket-Objekt ebenfalls die Methoden *Open* beziehungsweise *Close* an, wobei die dabei ausgeführten Operationen jedoch voneinander abweichen. Beim Aufruf von *Open* wird dabei die Socket-Verbindung zunächst anhand der zuvor beschriebenen Eigenschaften initialisiert, bevor die Socket-Verbindung in einem Empfangsmodus geöffnet wird. Dabei wird jedoch keine vollständige Verbindung zu einem Client hergestellt, sondern lediglich auf eingehende Anforderungen gewartet.

Bei eingehenden Anforderungen, Verbindungsauf- beziehungsweise Verbindungsabbau, Lesen oder Schreiben von Daten und eventuell auftretenden Fehlern werden auch vom TServerSocket-Objekt entsprechende Ereignisse ausgelöst, auf die im Rahmen der dazugehörigen Behandlungsroutinen reagiert werden kann. Die wichtigsten Ereignisse werden nachfolgend kurz erläutert [BOR01]:

- **OnListen(Sender: TObject; Socket: TCustomWinSocket)**: Dieses Ereignis tritt ein, bevor der Server-Socket für den Empfang geöffnet wird.

- **OnAccept(Sender: TObject; Socket: TCustomWinSocket)**: Das Auslösen dieses Ereignisses erfolgt nach dem Akzeptieren einer von einem Client gestellten Verbindungsanforderung.

- **OnClientConnect(Sender: TObject; Socket: TCustomWinSocket)**: Wenn ein Client-Socket eine vom Server-Socket akzeptierte Verbindung einrichtet, dann tritt das hier beschriebene Ereignis ein.

- **OnClientDisconnect(Sender: TObject; Socket: TCustomWinSocket)**: Dieses Ereignis wird ausgelöst, wenn eine der bestehenden Verbindungen zu einem Client-Socket geschlossen wird.

- **OnClientError(Sender: TObject; Socket: TCustomWinSocket; ErrorEvent: TErrorEvent; var ErrorCode: Integer)**: Wenn beim Einrichten, Verwenden oder Beenden der Socket-Verbindung zu einem Client-Socket ein Fehler auftritt, dann tritt das hier beschriebene Ereignis ein.

- **OnClientRead(Sender: TObject; Socket: TCustomWinSocket)**: Dieses Ereignis tritt ein, wenn der Server-Socket Informationen von einem Client-Socket lesen soll. Innerhalb der zugehörigen Behandlungsroutine müssen also die vom Parameter *Socket* bereitgestellten Methoden zum Lesen der vorhandenen Daten aufgerufen werden.
- **OnClientWrite(Sender: TObject; Socket: TCustomWinSocket)**: Ein Auslösen dieses Ereignisses erfolgt, wenn der Server-Socket Informationen an einen Client-Socket sendet.

Die vom Socket-Objekt bereitgestellten Methoden zum Lesen und Schreiben von Daten aus der beziehungsweise in die Socket-Verbindung sind mit denen des Socket-Objektes vom TClientSocket identisch. Außerdem stellt das Socket-Objekt auch die Eigenschaft *Data* zur Verfügung, welche auf anwendungsspezifische Daten für den Socket verweist und somit eine Verbindung von Informationen zur Socket-Verbindung mit dem entsprechenden Windows-Socket-Objekt erlaubt. So können beispielsweise Identifizierungsinformationen in dieser Eigenschaft gespeichert werden, die ein Server-Socket für die Auswertung von Client-Verbindungsanforderungen verwenden kann [BOR01].

4.2 Benutzer-, Gruppen- und Dienstverwaltung

Das im *Kapitel 3.1* beschriebene Konzept zum Sitzungsmanagement bildet die Grundlage für die Verwaltung der an das System angemeldeten Benutzer und der von ihnen verwendeten Dienste. Zur Umsetzung dieser Konzeption werden mehrere Klassen verwendet, deren Strukturen begleitend zur Beschreibung des Programmablaufes nachfolgend erläutert werden. Dabei werden auch Auszüge aus dem Quelltext dargestellt, die allerdings an einigen Stellen anstatt der realen Programmanweisungen zur kompakteren Darstellung entsprechende Verweise in Form von Kommentaren enthalten. Im GWOTS verwendete Klassen, die lediglich zur grafischen Darstellung dienen oder nicht im direkten Zusammenhang mit den Verwaltungs- und Kommunikationabläufen stehen, werden an dieser Stelle aus Gründen der Übersichtlichkeit nicht weiter betrachtet. Bevor jedoch die Implementierungen zur verwendeten Datenbank und deren Server-seitiger Verwaltung erläutert werden, erfolgen zunächst die Vorstellung des entworfenen Sitzungsmanagements und die Beschreibung der damit verbundenen Programmabläufe.

4.2.1 Sitzungsmanagement

Die Klasse *TfrmSvMain* stellt die eigentliche Hauptklasse im hier entwickelten Server-Modul dar. Sie nimmt dabei unter anderem die aus dem nachfolgenden Auszug des *Quelltextes 4.1* ersichtlichen Klassen des Sitzungsmanagements, Kommunikation via Socket und der einzelnen

Dienst-Server-Module auf, welche im Folgenden näher vorgestellt werden.

```
TfrmSvMain = class(TForm)
   ...
  {Objekt zum Sitzungsmanagement}
  SessionMgr: TSessionMgr;
  {Server Socket−Objekt}
  tcpSvSockCtrl : TServerSocket;
  {Objekt zur Verwaltung der Nutzerdaten der Datenbank}
  fraUserAdmin: TfraUserAdmin;
  {Objekte der einzelnen Dienst−Server−Module}
  fraChat : TfraChat;
  fraWB: TfraWB;
  fraConf : TfraConf;
private
   ...
public
   ...
end;
```

Quelltext 4.1: Die Klasse *TfrmSvMain* und darin enthaltene Objekte

Beim Start des Server-Moduls werden zunächst die Objekte der im *Quelltext 4.1* aufgeführten Klassen erzeugt und initialisiert. Danach stellt das TSessionMgr-Objekt alle Funktionalitäten bereit, die zur Verwaltung von Gruppen und Nutzern im Rahmen der angebotenen Dienste erforderlich sind. Hierzu verwendet das TSessionMgr-Objekt weitere Objekte der Klassen *TUser*, *TGroup* und *TBasicGroup*, welche ebenfalls nach ihrer Erzeugung mit Standardwerten initialisiert werden. Diese Objekte stellen ihrerseits alle Eigenschaften und Methoden zur Verfügung, um Gruppen beziehungsweise Nutzer im System abzubilden. Das Zusammenspiel zwischen den hier beschriebenen Objekten wird aus der *Abbildung 4.1* ersichtlich und im Folgenden anhand des Programmablaufes genauer beschrieben.

Nach der Anmeldung eines Nutzers an das System wird für diesen ein TUser-Objekt erzeugt, welches daraufhin initialisiert wird. In der *Abbildung 4.1* werden beispielhaft zwei dieser Objekte dargestellt, wobei diese die Bezeichnung *User1* und *User2* tragen. Bei der Initialisierung dieser Objekte werden neben der Nutzerkennung des Anwenders auch die Berechtigungen für die angebotenen Dienste vermerkt und das zur Kommunikation vom Server eingerichtete Socket-Objekt mit dem TUser-Objekt verknüpft. Während die Eigenschaften zur Nutzungsberechtigung der Dienste in *Abbildung 4.1* nicht abgebildet sind, werden die Eigenschaften für die Nutzerkennung und die Socket-Objekte anhand der Bezeichnungen *Login* beziehungsweise *CtrlSocket*, *ChatSocket* und *WbSocket* innerhalb der abgebildeten TUser-Objekte dargestellt. *CtrlSocket* beinhaltet dabei die Adresse des Socket-Objektes vom zentralen Server (vgl. *Abbildung 3.3*), *ChatSocket* und *WbSocket* entsprechend die Adressen der Dienst-Sockets für Chat und Whiteboard. Das TUser-Objekt wird außerdem zum Basisdienst zugeordnet, indem es der im TBasicGroup-Objekt enthaltenen Nutzerliste hinzugefügt wird. In der *Abbildung 4.1* wird

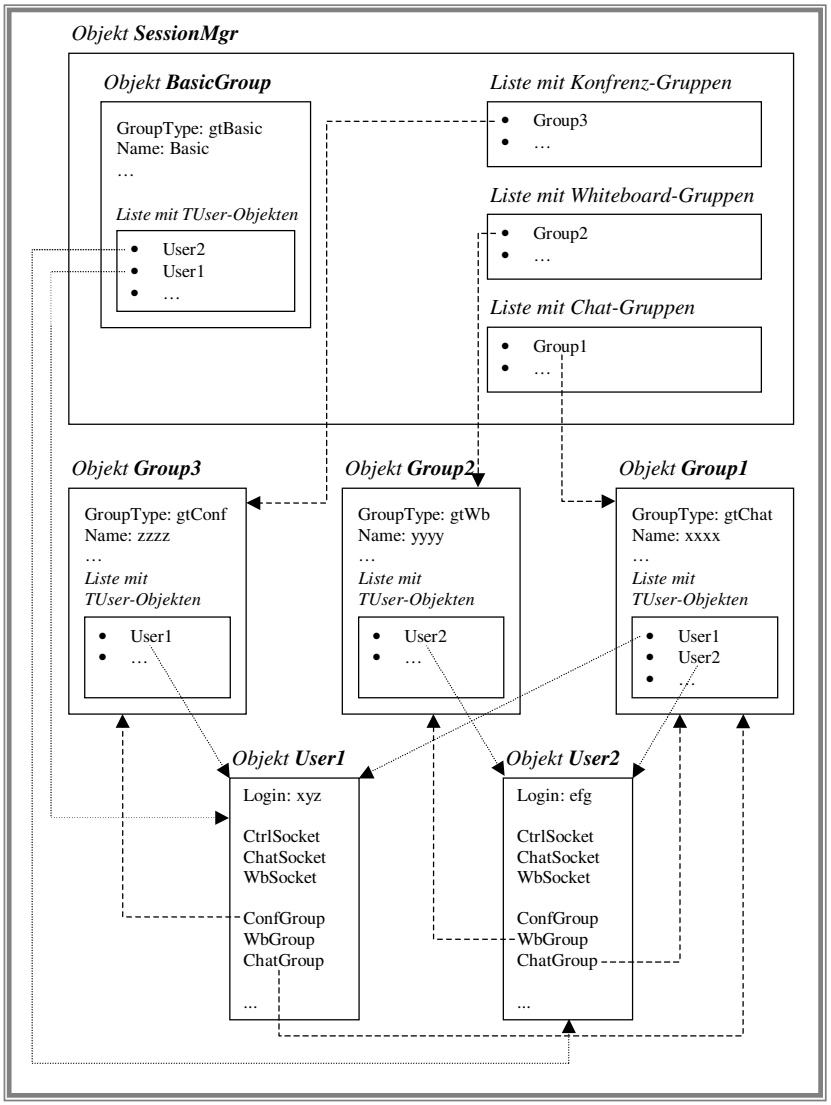

Abbildung 4.1: Schematische Darstellung des Sitzungsmanagements

diese Zuordnung durch die Verbindung der Objekte *BasicGroup* und *User1* beziehungsweise *User2* anhand gepunkteter Linien kenntlich gemacht, wobei das Objekt *BasicGroup* vom Typ TBasicGroup ist. Das TBasicGroup-Objekt beinhaltet somit eine Übersicht und die Zugriffsmöglichkeit für alle am System angemeldeten Nutzer. Ein in der Basisgruppe enthaltenes TUser-Objekt bleibt solange in dieser erhalten, bis sich der Nutzer vom System abmeldet. Der Beitritt in eine Gruppe eines Zusatzdienstes beziehungsweise das Verlassen selbiger hat dabei auf die Basisgruppe keinerlei Einfluss.

Bei der Eröffnung einer Gruppe für die Zusatzdienste Chat, Whiteboard beziehungsweise Videokonferenz durch einen am Basisdienst angemeldeten Nutzer wird ein entsprechendes Objekt der Klasse *TGroup* erzeugt und der entsprechenden Gruppenliste des TSessionMgr-Objektes, welches in *Abbildung 4.1* die Bezeichnung *SessionMgr* trägt, hinzugefügt. So-

mit kann über die betreffende Gruppenliste auf alle existierenden Gruppen eines Zusatzdienstes zugegriffen werden. In der *Abbildung 4.1* wird für jede dieser Zusatzdienstarten jeweils ein TGroup-Objekt dargestellt, wobei diese die Bezeichnung *Group1*, *Group2* beziehungsweise *Group3* tragen. Die Zuordnung dieser Objekte zur entsprechenden Gruppenliste des TSessionMgr-Objektes wird dabei durch gestrichelte Linien kenntlich gemacht wird, welche von der jeweiligen Gruppenliste zu den zugehörigen TGroup-Objekten führen. Dem TGroup-Objekt wird zudem der bei der Gruppeneröffnung angegebene Gruppenname sowie die Adresse des zum Nutzer gehörigen TUser-Objektes in der dafür vorgesehenen Nutzerliste zugewiesen, um über das TGroup-Objekt auf die zur jeweiligen Gruppe gehörenden Nutzer zugreifen zu können. In *Abbildung 4.1* wird diese Zuordnung durch gepunktete Linien verdeutlicht, welche von den dort abgebildeten TGroup-Objekten *Group1*, *Group2* und *Group3* auf die TUser-Objekte *User1* beziehungsweise *User2* verweisen. Analog hierzu wird auch im TUser-Objekt die Adresse des Gruppen-Objektes zum jeweiligen Dienst vermerkt, um über ein Nutzer-Objekt auf die von ihm verwendeten Gruppen-Objekte zugreifen zu können. Diese Zuordnung ist ebenfalls in der *Abbildung 4.1* dargestellt und wird durch die Verbindung der im TUser-Objekt vorhandenen Gruppeneigenschaften *ConfGroup*, *WbGroup* beziehungsweise *ChatGroup* zum jeweiligen Gruppenobjekt *Group1*, *Group2* beziehungsweise *Group3* anhand gestrichelter Linien kenntlich gemacht.

Tritt ein Nutzer einer bereits bestehenden Gruppe eines der hier beschriebenen Zusatzdienste bei, so erfolgt die Zuordnung zwischen dem Nutzer und dem in diesem Fall bereits existierenden TGroup-Objekt auf die gleiche Art und Weise, wie es bei der Eröffnung einer solchen Gruppe praktiziert wird.

Verlässt ein Nutzer die Gruppe eines der Zusatzdienste, so wird die Adresse des zugehörigen TUser-Objektes aus der Nutzerliste des Gruppenobjektes entfernt. Analog hierzu wird auch beim TUser-Objekt die Adresse des zum betroffenen Dienst gehörigen Gruppenobjektes zurückgesetzt. Die Zuordnung von Nutzer und Zusatzdienst-Gruppe wird somit aufgehoben. Tritt dabei der Fall auf, dass die betroffene Gruppe nach dem Verlassen eines Nutzers keine Mitglieder mehr beinhaltet, so wird das zugehörige TGroup-Objekt aus der entsprechenden Dienstliste des TSessionMgr-Objektes entfernt und anschließend freigegeben.

Beim Abmelden des Nutzers vom System wird dieser zunächst aus allen Zusatzdiensten entfernt, sofern er diese zu diesem Zeitpunkt noch nicht verlassen hat. Danach wird die Adresse des zugehörigen TUser-Objektes auch aus der Nutzerliste des Basisdienstes entfernt und das Nutzerobjekt freigegeben. Die zuvor beschriebene Auflösung einer Gruppe, verbunden mit der Freigabe des zugehörigen Gruppenobjektes, gilt nur für die Gruppen von Zusatzdiensten, nicht jedoch für die hier betrachtete Basisgruppe. Das TBasicGroup-Objekt bleibt immer erhalten, auch wenn es keine Benutzerobjekte beinhaltet.

Nach der Darstellung des Zusammenspiels der zum Sitzungsmanagement verwendeten Klassen *TUser*, *TGroup*, *TBasicGroup* und *TSessionMgr* werden deren Implementierungen

sowie weitere damit verbundene Vorgänge bezüglich des Programmablaufes im Detail vorgestellt.

Die Klasse *TUser*

Die Klasse *TUser* dient zur Abbildung eines Nutzers auf das System. Sie stellt neben den bereits erwähnten Schreib/Lese-Eigenschaften, wie beispielsweise der Nutzerkennung oder der Nutzungsberechtigung für die einzelnen Dienste, noch weitere zur Verfügung, welche aus dem *Quelltext 4.2* hervorgehen.

```
TUser = class (TObject)
 private
   ...
 public
   {Nutzerkennung und IP−Adresse des Client−Rechners}
   property Login: String read FLogin write FLogin;
   property IPAddr: String read FIPAddr write FIPAddr;
   {Datenbank−Indizes fuer Matrikel , Fachbereich und Chat}
   property RegID: Integer read FRegID write FRegID;
   property DeptID: Integer read FDeptID write FDeptID;
   property ChatID: Byte read FChatID write FChatID;
   {Kennzeichnung, ob Nutzer an einen der aufgefuehrten Dienste angemeldet ist}
   property InChat: Boolean read FInChat write FInChat;
   property InWb: Boolean read FInWb write FInWb;
   property InConf: Boolean read FInConf write FInConf;
   {Kennzeichnung der Dienste , die dem Nutzer zur Verfuegung stehen}
   property CanFtp: Boolean read FCanFtp write FCanFtp;
   property CanMail: Boolean read FCanMail write FCanMail;
   property CanConf: Boolean read FCanConf write FCanConf;
   property CanChat: Boolean read FCanChat write FCanChat;
   property CanWb: Boolean read FCanWb write FCanWb;
   property CanLogin: Boolean read FCanLogin write FCanLogin;
   {Anmeldezeit des Nutzers}
   property LogTime: TDateTime read FLogTime write FLogTime;
   {Steuer− und Dienst−Sockets des Nutzer−Client}
   property CtrlSocket : TCustomWinSocket read FCtrlSocket write FCtrlSocket;
   property ChatSocket: TCustomWinSocket read FChatSocket write FChatSocket;
   property WbSocket: TCustomWinSocket read FWbSocket write FWbSocket;
   {Zugriff auf die Gruppenobjekte eines Nutzers}
   property ChatGroup: TGroup read FChatGroup write SetChatGroup;
   property WbGroup: TGroup read FWbGroup write SetWbGroup;
   property ConfGroup: TGroup read FConfGroup write SetConfGroup;
   {zur Steuerung des Datenaustauschs verwendet}
   property CtrlRec : TRecType read FCtrlRec write FCtrlRec;
end;
```

Quelltext 4.2: Eigenschaften der Klasse *TUser*

Hierzu zählen vor allem die Eigenschaften *ChatGroup*, *WbGroup* sowie *ConfGroup*, welche die Adressen der TGroup-Objekte der korrespondierenden Zusatzdienste beinhalten und

somit den Zugriff auf die vom Nutzer beanspruchten Gruppen erlauben. Dieser Zugriff erfolgt beispielsweise immer dann, wenn beim entsprechenden Dienst-Server-Modul Daten für Chat oder Whiteboard eintreffen und von diesem innerhalb der zugehörigen Gruppe weitergeleitet werden müssen. Beim Eintreffen solcher Chat- oder Whiteboard-Daten erfolgt hierbei zunächst der Zugriff auf das Nutzerobjekt des Absenders der Daten anhand des empfangenden Socket-Objektes, welches bei der Anmeldung eines Nutzers an einen Dienst mit der Adresse des Nutzerobjektes verknüpft wird, bevor über die zum TUser-Objekt gehörige Eigenschaft *ChatGroup* beziehungsweise *WbGroup* der Zugriff auf das entsprechende Gruppenobjekt erfolgt. Dieses beinhaltet, wie aus *Abbildung 4.1* ersichtlich, eine Liste aller Adressen der zugehörigen TUser-Objekte und liefert beim Auslesen deren Eigenschaften *ChatSocket* beziehungsweise *WbSocket* die Adressen der Socket-Objekte, über welche die zu verteilenden Daten vom Dienst-Server-Modul gesendet werden müssen. Der nachfolgende Auszug aus dem *Quelltext 4.3* zeigt den dabei auftretenden Programmablauf und die Verwendung der beschriebenen Eigenschaften des TUser-Objektes am Beispiel des *OnClientRead*-Ereignisses des *TServerSocket*-Objektes beim Chat-Server.

```
procedure TfraChat.tcpSvSockChatClientRead(Sender: TObject; Socket: TCustomWinSocket);
var
  ...
begin
  User := TUser(Socket.Data);
  if (User = nil) then begin   {neuer Nutzer betritt Gruppe}
    Socket.ReceiveBuf(ServiceRec,SizeOf(ServiceRec));
    User := ServiceRec.UserAddr;
    {Nutzeradresse an Socket binden und umgekehrt}
    Socket.Data := User;
    User.ChatSocket := Socket;
    if (ServiceRec.Group <> nil) then
      {Nutzer zu bestehender Chat-Gruppe hinzufuegen}
      TGroup(ServiceRec.Group).AddUser(User)
    else
      {neue Chat-Gruppe erzeugen und Nutzer hinzufuegen}
      FSessionMgr.AddChatGroup(ServiceRec.GroupName).AddUser(TUser(ServiceRec.UserAddr));
  end
  else begin  {Chat-Nachricht empfangen und weiterleiten}
    ChatMsg := Socket.ReceiveText;
    {Nachricht an alle Mitglieder der Gruppe weiterleiten}
    for i:=User.ChatGroup.CountUser-1 downto 0 do begin
      {Senden der User-ID innerhalb der Chat-Gruppe zur Zuordnung des Absenders auf Client-Seite}
      User.ChatGroup.User[i].ChatSocket.SendBuf(User.ChatID,SizeOf(User.ChatID));
      {Senden der Nachricht}
      User.ChatGroup.User[i].ChatSocket.SendText(ChatMsg);
    end;
  end;
end;
```

Quelltext 4.3: Programmablauf beim Empfang von Daten am Beispiel des Chat-Servers

Die Klassen *TGroup* und *TBasicGroup*

Die Klassen *TGroup* und *TBasicGroup* dienen zur Abbildung der Gruppen, welche vom Sitzungsmanager verwaltet werden müssen, auf das System. Während *TGroup* zur Gruppenabbildung der Zusatzdienste für Chat, Whiteboard und Videokonferenz ausgelegt ist, wird die Klasse *TBasicGroup* zur Verwaltung der Nutzer des Hauptdienstes verwendet. *TBasicGroup* ist eine von *TGroup* abgeleitete Klasse, die sich von dieser lediglich in den Routinen zur Abmeldung eines Nutzern von einem Dienst unterscheidet. Während beim Abmelden eines Nutzers vom Hauptdienst das zugehörige TUser-Objekt freigegeben wird, erfolgt beim Verlassen eines Zusatzdienstes nur die Aufhebung der entsprechenden Verknüpfung zwischen Nutzerobjekt und Nutzerliste des jeweiligen Zusatzdienstes. Der nachfolgende *Quelltext 4.4* zeigt zunächst den Aufbau der Klasse *TGroup* mit ihren wichtigsten Eigenschaften und Methoden.

```
TGroup = class(TObject)
 private
   FUserList      : TList;     {Liste mit TUser-Objekten}
   FOnAddUser  : TAddUserEvent;
   FOnRemoveUser: TRemoveUserEvent;
    ...
 public
    constructor Create;
    destructor Free;
   {Listen zum Abspeichern von Whiteboard-Aktionen}
   WbDrawRecList: TList;
   WbTextRecList: TStrings;
   {Nutzerobjekt zur Gruppe hinzufuegen bzw. aus Gruppe entfernen}
   procedure AddUser(User: TUser);
   procedure RemoveUser(User: TUser); overload;
   procedure RemoveUser(Index: Integer); overload;
   {alle Nutzerobjekte aus Nutzerliste entfernen}
   procedure Clear;
   {Festlegung und Abfrage des Gruppentyps}
   property Type: TGroupType read FType write SetGroupType;
   {Festlegung und Abfrage des Gruppennamens}
   property Name: String read FName write FName;
   {User-ID innerhalb einer Chat-Gruppe}
   property ChatID: Byte read FChatID write FChatID;
   {Festlegung und Abfrage des Zeitpunktes der Gruppenbildung}
   property StartTime: TDateTime read FStartTime write FStartTime;
   {Anzahl der zur Gruppe gehoerigen Nutzer}
   property CountUser: Integer read GetCountUser;
   {Zugriff auf Nutzerobjekt}
   property User[Index: Integer]: TUser read GetUser;
   {Ereignisroutinen ausloesen}
   property OnAddUser: TAddUserEvent read FOnAddUser write FOnAddUser;
   property OnRemoveUser: TRemoveUserEvent read FOnRemoveUser write FOnRemoveUser;
end;
```

Quelltext 4.4: Eigenschaften und Methoden der Klasse *TGroup*

Wie aus *Quelltext 4.4* hervorgeht, stellt diese Klasse unter anderem die Eigenschaft *Type* zur Verfügung, welche Auskunft über die zur Gruppe gehörigen Dienstart gibt. Diese Eigenschaft wird während der Erzeugung des Gruppenobjektes gesetzt und kann entsprechend des verwendeten Dienstes die aus *Quelltext 4.5* ersichtlichen Werte annehmen.

```
TGroupType = (gtBasic, gtChat, gtWb,gtConf);
```

Quelltext 4.5: Datentyp zur Kennzeichnung der Objekte *TGroup* und *TBasicGroup*

Die Operationen zur vorab beschriebenen Verknüpfung zwischen dem Nutzerobjekt und dem von der Dienstart abhängigen Gruppenobjekt werden im *Quelltext 4.6* dargestellt. Hieraus wird auch ersichtlich, dass beim Auslösen der zugehörigen Ereignisroutine die Adressen des Gruppen- und des Nutzerobjektes für weitere Aktionen bereitgestellt werden. Im vorliegenden System dient die hierbei ausgelöste Ereignisroutine speziell zur Aktualisierung der bereits angesprochenen Gruppen- und Nutzerlisten, die den Client-Anwendungen für die einzelnen Dienste bereitgestellt werden.

```
procedure TGroup.AddUser(User: TUser);
begin
  {Hinzufuegen der Adresse des TUser-Objektes zur Nutzerlis"-te der Gruppe}
  FUserList.Add(User);
  {Zuweisung der Adresse des Gruppenobjektes zum Nutzerobjekt auf Nutzerseite}
  case Typ of
    gtChat:
    begin
      User.ChatGroup := Self;
      User.InChat := True;
      {Zuweisung der innerhalb der Chat-Gruppe ügltigen User-ID}
      User.ChatID := ChatID mod 256;
      inc(ChatID);
    end;
    gtWb:
      ...
    gtConf:
      ...
  end;
  {Ausloesen der zugehoerigen Ereignisroutine}
  if Assigned(FOnAddUser) then FOnAddUser(Self,User);
end;
```

Quelltext 4.6: Methode *AddUser* der Klasse *TGroup*

Die Implementierung zur Aufhebung der angesprochenen Verknüpfungen zwischen einem Nutzerobjekt und dem zugehörigen Gruppenobjekt ist Gegenstand des im *Quelltext 4.7* dargestellten Programmcodes. Hierbei wird zunächst die Adresse des Nutzerobjektes aus der zugehörigen Nutzerliste entfernt, bevor die Verknüpfung zum Gruppenobjekt über die korrespondierende Eigenschaft des TUser-Objektes zurückgestzt wird. Beim Auslösen der hierzu

vorhandenen Ereignisroutine wird neben den Adressen des Gruppen- und Nutzerobjektes noch ein dritter Parameter übergeben, welcher Auskunft darüber gibt, ob der entsprechende Nutzer die Gruppe eines Zusatzdienstes verlässt, oder ob er sich vom gesamten System abmeldet.

```
procedure TGroup.RemoveUser(User: TUser);
var
  Index : Integer;
begin
  {Adresse des Nutzerobjektes aus Gruppenliste entfernen}
  Index := FUserList.IndexOf(User);
  if ((Index > -1) and (Index < FUserList.Count)) then FUserList.Delete(Index);
  {Aufheben der Verknuepfung zwischen Gruppenobjekt und Nutzerobjekt auf Nutzerseite}
  case Typ of
    gtChat:
    begin
      User.ChatGroup := nil;
      User.InChat := False;
    end;
    gtWb : ...
    gtConf : ...
  end;
  {Ausloesen der zugehoerigen Ereignisroutine}
  if Assigned(FOnRemoveUser) then FOnRemoveUser(Self,User,False);
end;
```

Quelltext 4.7: Methode *RemoveUser* der Klasse *TGroup*

Die Klasse *TBasicGroup* stellt lediglich eine Ableitung der Klasse *TGroup* dar, welche beim Entfernen eines Nutzers aus der Gruppe hier zusätzlich auch das Nutzerobjekt selbst freigibt. Zu diesem Zweck werden die betreffenden Methoden neu vorgestellt, wie es aus dem *Quelltext 4.8* hervorgeht. Die hierbei abgebildete Methode *RemoveUser* verdeutlicht den Unterschied zur gleichnamigen Methode der Basisklasse *(vgl. Quelltext 4.7)*.

```
TBasicGroup = class(TGroup)
public
  destructor Free;
  { Listeneintrag aus Hauptgruppe entfernen und Nutzerobjekt freigeben}
  procedure RemoveUser(User: TUser); reintroduce ; overload;
  { Listeneintrag aus Hauptgruppe entfernen und Nutzerobjekt freigeben}
  procedure RemoveUser(Index: Integer); reintroduce ; overload;
  {alle Listeneintraege der Hauptgruppe entfernen und Nutzerobjekte freigeben}
  procedure Clear; reintroduce ;
end;

procedure TBasicGroup.RemoveUser(User: TUser);
  ...
  {Ausloesen der Ereignisroutine}
  if Assigned(FOnRemoveUser) then FOnRemoveUser(Self,User,True);
  {Adresse des Nutzerobjektes aus Nutzerliste entfernen}
  FUserList.Delete(Index);
```

```
    {Freigabe des Nutzerobjektes}
    User.Free;
  end;
```

Quelltext 4.8: Eigenschaften und Methoden der Klasse *TBasicGroup*

Die Klasse *TSessionMgr*

Das in *Abbildung 4.1* dargestellte Objekt *SessionMgr* wird durch die Klasse *TSessionMgr* beschrieben. Diese bildet die Basis für die Verwaltung der vom zentralen Dienst sowie den Zusatzdiensten benötigten Nutzer- und Gruppeninformationen und wird im *Quelltext 4.9* auszugsweise dargestellt.

```
TSessionMgr = class(TObject)
 private
   {Hauptgruppe mit allen angemeldeten Nutzern}
   FBasicGroup    : TBasicGroup;
   {Listen mit TGroup-Objekten fuer die einzelnen Dienste}
   FWbGroups,
   FChatGroups,
   FConfGroups    : TList;
   { Ereignisroutinen }
   FOnAddGroup : TAddGroupEvent;
   FOnRemoveGroup: TRemoveGroupEvent;
    ...
   { liefert eine Gruppe anhand ihres Namen aus der angegebenen Liste zurueck}
   function GetGroup(GroupName: String; GroupList: TList ): TGroup;
   {Freigabe aller Objekte von GroupList inkl . GroupList selbst }
   procedure ClearGroupList(GroupList: TList );
 public
    constructor Create;
    destructor Free;
   {Hinzufuegen von Zusatzdienst-Gruppen in den Sitzungsmanager}
   function AddChatGroup(GroupName: String): TGroup;
   function AddWbGroup(GroupName: String): TGroup;
   function AddConfGroup(GroupName: String): TGroup;
   {Entfernen von Zusatzdienst-Gruppen aus dem Sitzungsmanager}
   procedure RemoveChatGroup(Group: TGroup);
   procedure RemoveWbGroup(Group: TGroup);
   procedure RemoveConfGroup(Group: TGroup);
   {Anzahl der existierenden Zusatzdienst-Gruppen fuer die einzelnen Dienste}
   property CountWbGroups: Integer read GetCountWbGroups;
   property CountChatGroups: Integer read GetCountChatGroups;
   property CountConfGroups: Integer read GetCountConfGroups;
   {Zugriff auf alle vom Sitzungsmanager verwalteten Gruppen}
   property BasicGroup: TBasicGroup read FBasicGroup;
   property WbGroup[Index: Integer]: TGroup read GetWbGroup;
   property ChatGroup[Index: Integer ]: TGroup read GetChatGroup;
   property ConfGroup[Index: Integer ]: TGroup read GetConfGroup;
   { Ereignisroutinen ausloesen}
   property OnAddGroup: TAddGroupEvent read FOnAddGroup write FOnAddGroup;
```

```
    property  OnRemoveGroup: TRemoveGroupEvent read FOnRemoveGroup write FOnRemoveGroup;
end;
```

Quelltext 4.9: Eigenschaften und Methoden der Klasse *TSessionMgr*

Die von *TSessionMgr* angebotenen Eigenschaften *BasicGroup*, *WbGroup*, *ChatGroup* und *ConfGroup* bieten hierbei Zugriff auf das in dieser Klasse enthaltene TBasicGroup-Objekt beziehungsweise die für die Zusatzdienste verwendeten und mit Hilfe von Listen-Objekten verwalteten TGroup-Objekte sowie deren Eigenschaften und Methoden. Die Methoden zum Hinzufügen von Zusatzdienst-Gruppen fügen bei der Eröffnung einer solchen Gruppe deren Objektadresse dem mit der Dienstart korrespondierenden Listen-Objekt hinzu. Analog hierzu entfernen die Methoden zur Auflösung einer Gruppe die zugehörige Objektadresse aus der entsprechenden Dienstliste und geben danach das betroffene Gruppenobjekt selbst frei.

Beim Hinzufügen oder Entfernen von Gruppen werden analog dem Hinzufügen oder Entfernen von Nutzern Ereignisroutinen ausgelöst, welche ebenfalls zur Aktualisierung der bereits angesprochenen Gruppen- und Nutzerlisten dienen, die dem Anwender innerhalb der Client-Anwendung für die einzelnen Dienste zur Verfügung stehen.

Datenstrukturen zur Client/Server-Kommunikation

Der Informationsaustausch zwischen Client und Server erfolgt beim GWOTS durch das Versenden und Empfangen von strukturierten Datentypen, welche unter Delphi als *Records* bezeichnet werden. Die im Rahmen der vorliegenden Arbeit definierten Records, welche zur An- beziehungsweise Abmeldung eines Nutzers an beziehungsweise von einem Dienst Verwendung finden, werden nachfolgend kurz beschrieben. Als Erstes wird hierbei der Record *TClLoginRec* vorgestellt, welcher bei der Nutzeranmeldung von der Client-Anwendung an den Server verschickt wird. Dieser Record beinhaltet neben der Nutzerkennung und dem Passwort die von der Client-Maschine ermittelten Ressourcen, welche beim Server ausgewertet werden müssen.

```
TClLoginRec = record
   Login,
   Passwd:  String [20];
   {Ressourcen−Informationen des Client−Rechners}
    ...
end;
```

Quelltext 4.10: Der Record *TClLoginRec*

Als Antwort auf eine erfolgreiche Anmeldung an das System sendet der Server den im *Quelltext 4.11* dargestellten Record. Dieser beinhaltet neben der Start-URL der Vorlesungsinhalte, der IP-Adresse des Videokonferenz-Servers und den Berechtigungen zur Nutzung der Zusatzdienste auch die Port-Nummern der Server-Sockets für den Chat-Dienst und den Whi-

teboard-Dienst, so dass diese bei diesbezüglich eventuell auftretenden Änderungen am Dienst-Server-Modul automatisch auf der Client-Seite angepasst werden. Außerdem wird auch die Adresse des vom Server verwalteten, zugehörigen TUser-Objektes übermittelt, da diese zur Anmeldung an den Chat-Dienst beziehungsweise Whiteboard-Dienst benötigt wird. Dies begründet sich dadurch, dass der Sitzungsmanager vom zentralen Client und den Dienst-Server-Modulen für Chat und Whiteboard gemeinsam verwendet wird. Bei der Anmeldung eines Nutzers an einen Zusatzdienst existiert somit das zum Nutzer gehörige TUser-Objekt bereits, so dass es nicht nochmals erzeugt werden darf. Stattdessen erfolgt in einem solchen Fall die Zuordnung vom Nutzerobjekt zu dem entsprechenden Socket-Objekt Server-seitig über die angesprochene Adresse, die zu diesem Zweck bei der Anmeldung an den Dienst vom Client an das Dienst-Server-Modul übertragen werden muss *(vgl. Quelltext 4.3)*.

```
TSvInitRec = record
  UserAddr                              : Pointer;
  CanFtp,CanMail,CanConf,CanChat,CanWb: Boolean;
  StartURL                              : String [255];
  ConfServer                            : String [16];
  WbSvPort,ChatSvPort                   : Integer;
end;
```

Quelltext 4.11: Der Record *TSvInitRec*

Zur Anmeldung beziehungsweise Abmeldung an einen Zusatzdienst wird der im *Quelltext 4.12* beschriebene Record *TServiceRec* verwendet. Dieser beinhaltet neben der zuvor bereits erläuterten Adresse des TUser-Objektes, welche Server-seitig zur Verknüpfung von Nutzerobjekt und Socket-Objekt beziehungsweise Nutzerobjekt und Gruppenobjekt verwendet wird, auch die Adresse beziehungsweise den Namen der Gruppe, welcher der Anwender beitreten möchte. Wählt der Anwender die Gruppe aus der ihm zur Verfügung stehenden Gruppenliste aus, so wird die mit dem Gruppennamen zusammen vermerkte Adresse des Gruppenobjektes dem Record zugewiesen. Bei der Neueröffnung einer Gruppe durch den Anwender hingegen wird im Record lediglich der Gruppenname vermerkt. Die im Record zu vermerkende Adresse des Gruppenobjektes wird in diesem Fall mit dem Null-Wert belegt, so dass das Dienst-Server-Modul die Eröffnung einer neuen Gruppe erkennen kann und dementsprechende Aktionen ausführt.

```
TServiceRec = record
  UserAddr : Pointer;
  Group    : Pointer;
  GroupName: String[20];
end;
```

Quelltext 4.12: Der Record *TServiceRec*

Zur Aktualisierung der auf Seiten des Client vorhandenen Gruppen- und Nutzerlisten findet der Record *TListRec* Verwendung, dessen Struktur im *Quelltext 4.13* abgebildet ist. Neben den Feldern *Str*, *Obj* und *Group*, welche eine Nutzerkennung beziehungsweise den Namen einer Gruppe und hierzu korrespondierende Objektadressen beinhalten, ist auch ein Feld namens *Typ* in der Record-Struktur enthalten. Dieses Feld übermittelt dem Client, ob die im Record enthaltenen Informationen den Beitritt oder Austritt eines Nutzers in beziehungsweise aus einer Gruppe beschreiben, oder ob hiermit die Neueröffnung oder Auflösung einer Zusatzdienst-Gruppe bekanntgegeben wird. Zu diesem Zweck dient der Datentyp *TListRecType*, welcher die hierbei möglichen Vorgänge bezeichnet und ebenfalls im *Quelltext 4.13* dargestellt wird.

```
TListRecType = (lrAddUser,lrAddUserList,lrRemoveUser,lrAddChat,lrRemoveChat,
                lrAddWb,lrRemoveWb,lrAddConf,lrRemoveConf,lrAddUserChat,
                lrRemoveUserChat,lrAddUserWb,lrRemoveUserWb,lrAddUserConf,
                lrRemoveUserConf);


TListRec = record
  Typ: TListRecType;
  Str: String [20];
  Obj: Pointer;
  Grp: Pointer;
end;
```

Quelltext 4.13: Der Record *TListRec* und der Datentyp *TListRecType*

Kapselung der Dienst-Server-Routinen

Die im *Quelltextauszug 4.1* aufgeführten Klassen *TfraChat*, *TfraWb* und *TfraConf* dienen zur Kapselung der Server-Routinen für die Zusatzdienste Chat, Whiteboard beziehungsweise Videokonferenz, wobei sich die Routinen des Videokonferenzdienstes hierbei auf den Austausch von Steuernachrichten zur Umsetzung des in den *Kapiteln 3.1* und *3.3.3* beschriebenen einheitlichen Sitzungsmanagements beschränken. Eine nähere Vorstellung der in diesen Server-Klassen integrierten Routinen und den damit verbundenen Ablaufen beim Austausch von Informationen erfolgt im Rahmen der Beschreibungen zur Implementierung dieser Zusatzdienste in den *Kapiteln 4.4.2, 4.4.3* beziehungsweise *4.4.4*.

4.2.2 Datenverwaltung

Im *Kapitel 3.1.2* wird die Konzeption der vom GWOTS verwendeten Datenbank zur Verwaltung der Nutzerdaten beschrieben. Das dabei vorgestellte RDM wird im hier entwickelten System für eine MS-Access Datenbank implementiert. Um die Konsistenz der darin enthaltenen Daten sicherstellen zu können, wird für die Fremdschlüsselbeziehungen unter den einzelnen Relationen festgelegt, dass diese die referentielle Integrität erfüllen müssen. Das bedeutet, dass

für einen Datensatz der Relation *Matrikel* im Attribut *FbID* ein Wert enthalten sein muss, der mit einem Attribut des Primärschlüssels der Relation *Fachbereich* korrespondiert. Gleiches gilt auch für die Fremdschlüsselbeziehung zwischen den Relationen *Matrikel* und *Nutzer*.

Außerdem wird zur Sicherstellung der referentiellen Integrität auch das kaskadierte Löschen von Datensätzen vorgeschrieben. Das hat zur Folge, dass alle Datensätze, die in einer Fremdschlüsselbeziehung zu einem Datensatz der hierbei korrespondierenden Relation stehen, bei dessen Entfernung aus der Datenbank ebenfalls gelöscht werden. Wird also beispielsweise ein bestimmter Datensatz aus der Relation *Fachbereich* entfernt, so hat dies auch das Löschen aller von diesem abhängigen Datensätze der Relationen *Matrikel* und *Nutzer* zur Folge. Um ein versehentliches Löschen von Datensätzen zu verhindern, werden alle Löschoperationen vor ihrer Ausführung mit Hilfe eines Dialogfeldes angezeigt und müssen vom Administrator bestätigt werden, bevor sie tatsächlich ausgeführt werden.

Die verwendeten Datentypen für die Relation *Nutzer* werden aus der *Tabelle 4.1* ersichtlich. Login, Passwort, Vor- und Nachname sowie Matrikel müssen zwingend angegeben werden, um einen Datensatz in diese Relation einfügen zu können. Das Login muss dabei eindeutig sein.

Die Relation *Nutzer* mit ihren Attributen			
Feld	**Beschreibung**	**Feldtyp**	**Info**
Login	Kennung	VarChar(10)	Primärschlüssel
Passwort	Passwort	VarChar(10)	NOT NULL
Name	Nachname	VarChar(30)	NOT NULL
Vorname	Vorname	VarChar(30)	NOT NULL
MatrID	zugehöriges Matrikel	Integer	Fremdschlüssel
MatrNr	Matrikel-Nr.	VarChar(10)	max. 10 Zeichen
Mail	E-Mail-Adresse	VarChar(50)	max. 50 Zeichen
Basis	Login-Berechtigung	Bit	Sperrung $\widehat{=}$ 0
Chat	Berechtigung für Chat	Bit	Sperrung $\widehat{=}$ 0
Whiteboard	Berechtigung für WB	Bit	Sperrung $\widehat{=}$ 0

Tabelle 4.1: Struktur der Datenbanktabelle *Nutzer* zur Benutzerverwaltung

Zur Verwaltung der existierenden Matrikel wird die Relation *Matrikel* verwendet. Die bei der Implementierung benutzten Datentypen sind in der *Tabelle 4.2* aufgeführt. Als Index wird hier ein ganzzahliger Autoinkrement-Wert verwendet, wie es das Konzept aus *Kapitel 3.1.2* vorsieht. Der Matrikelname muss für jeden in dieser Tabelle enthaltenen Datensatz eindeutig sein.

Die Datentypen, welche für die Implementierung der Relation *Fachbereich* Verwendung finden, sind aus *Tabelle 4.3* ersichtlich. Auch hier wird ein ganzzahliger Autoinkrement-Wert als Primärschlüssel eingesetzt, und der Name des Fachbereiches muss ebenfalls innerhalb die-

ser Tabelle eindeutig sein.

Die Relation *Matrikel* mit ihren Attributen			
Feld	**Beschreibung**	**Feldtyp**	**Info**
MatrID	Index	Integer	Primärschlüssel
MatrName	Name des Matrikels	VarChar(10)	eindeutig
FbID	zugehöriger Fachbereich	Integer	Fremdschlüssel
StartURL	Startseite	VarChar(255)	max. 255 Zeichen

Tabelle 4.2: Struktur der Datenbanktabelle *Matrikel* zur Benutzerverwaltung

Die Relation *Fachbereich* mit ihren Attributen			
Feld	**Beschreibung**	**Feldtyp**	**Info**
FbID	Index	Integer	Primärschlüssel
FbName	Name des Fachbereichs	VarChar(50)	eindeutig

Tabelle 4.3: Struktur der Datenbanktabelle *Fachbereich* zur Benutzerverwaltung

Zur Verwaltung der Nutzerdaten ist im Server-Modul ein Dialogfenster integriert, welches sowohl die Einsicht als auch die Manipulation der Datenbankinhalte erlaubt. Die hier angesprochenen Funktionalitäten sind in der Klasse *TfraUserAdmin* gekapselt, welche ihrerseits in Form des Objektes *fraUserAdmin* Bestandteil der Server-Modul-Klasse *TfrmSvMain* ist *(vgl. Quelltext 4.1)*. Der nachfolgende Auszug des *Quelltextes 4.14* zeigt die Methoden dieser Klasse, welche nach außen hin zur Verfügung stehen.

```
uses  DataU  , ...;

TfraUserAdmin = class(TFrame)
    ...
 private
    ...
 public
   function  CheckLoginAndInitUser(LoginRec: TClLoginRec; var User: TUser): Boolean;
   function  GetStartURL(MatrID: Integer ):  String ;
end;
```

Quelltext 4.14: Die Klasse *TfraUserAdmin* und ihre öffentlichen Methoden

Die Funktion *CheckLoginAndInitUser* besitzt zwei Parameter. Als erster Parameter wird dieser Funktion der Login-Record übergeben, welcher von einem Client bei dessen Anmeldung an das System an den zentralen Server gesendet wird. Die darin enthaltenen Anmeldeinformationen, also Nutzerkennung und Passwort, werden innerhalb der Funktion mit den Daten der Datenbank verglichen. Fällt dieser Vergleich positiv aus, so gibt die Funktion den Wert TRUE zurück und das als zweiter Parameter übergebene TUser-Objekt wird anhand der zum Nutzer

gehörigen Datenbankdaten initialisiert. Sollte der Vergleich jedoch negativ ausfallen, so bedeutet dies, dass die Anmeldeinformationen nicht korrekt sind und der Rückgabewert der Funktion wird mit FALSE belegt. Bei Rückgabe des Wertes FALSE durch die Funktion *CheckLoginAndInitUser* sendet der zentrale Server eine entsprechende Fehlermeldung an den anfragenden Client und beendet danach sofort die Verbindung zu diesem.

Die zweite öffentliche Funktion *GetStartURL* liefert die Adresse der Startseite zum Vorlesungsinhalt entsprechend des zum Nutzer zugehörigen Matrikels. Als Parameter wird dieser Funktion der in der Datenbank vermerkte Matrikelindex übergeben, welcher nach erfolgreicher Initialisierung des TUser-Objektes durch die Funktion *CheckLoginAndInitUser* zu diesem Zweck vom Nutzerobjekt abgefragt wird.

Zum lesenden beziehungsweise schreibenden Zugriff auf die verwendete Datenbank stellt Borland Delphi™ unter anderem die Klassen *TDatabase* und *TQuery* zur Verfügung, welche im GWOTS in Verbindung mit der BDE eingesetzt werden. Bei der BDE handelt es sich um eine Windows-basierte Datenbank-Engine und Konnektivitäts-Software, welche eine Vielzahl von Datenbanktreibern beinhaltet [BOR01]. Diese hier angesprochenen Klassen werden im GWOTS mit Hilfe des Datenmodulobjektes *Data* integriert. Das Datenmodulobjekt wird in einer Unit namens *DataU* deklariert und dem Objekt *fraUserAdmin* durch Einbindung dieser Unit *(vgl. Quelltext 4.14)* zum Zugriff bereitgestellt. Der Aufbau der Datenmodulklasse und ihre öffentlichen Methoden werden auszugsweise im *Quelltext 4.15* dargestellt.

```
TData = class (TDataModule)
  Database : TDatabase;
  Query    : TQuery;
public
   ...
  procedure GetDBData(var TreeView: TTreeView; DataMode: TDataMode); overload;
  procedure InsertDBData(DataMode: TDataMode; UserData: TUserData);
  procedure UpdateDBData(DataMode: TDataMode; UserData: TUserData);
  procedure DeleteDBData(DataMode: TDataMode; ID: Integer; Value: String );
end;
```

Quelltext 4.15: Die Klasse *TData* und ihre wichtigsten Methoden

Die Methode *GetDBData* dient zur grafischen Darstellung der in der Datenbank enthaltenen Daten. Der erste Parameter *TreeView* beinhaltet dabei die Adresse eines Steuerelementes, welches der visuellen Präsentation der per SQL-Anweisung abgefragten Daten in Form einer hierarchischen Baumstruktur dient. Der zweite Parameter hingegen bestimmt den Inhalt der dabei gestellten SQL-Abfrage, welche folgende Zielstellung haben kann *(vgl. Abbildung 3.4)*:

- Auflistung aller Nutzernamen
- Auflistung aller Logins
- Auflistung aller Matrikel

- Auflistung aller zu einem bestimmten Matrikel gehörigen Nutzernamen
- Auflistung aller Fachbereiche
- Auflistung aller zu einem bestimmten Fachbereich gehörigen Nutzernamen
- Auflistung aller Daten, die einen bestimmten Nutzer betreffen.

Die Methoden *InsertDBData*, *UpdateDBData* und *DeleteDBData* dienen zum Einfügen, Ändern beziehungsweise Löschen von Datenbankinhalten, welche hierbei Nutzer-, Matrikel- oder Fachbereichsdaten betreffen. Der Inhalt der dabei auszuführenden SQL-Anweisung wird hierzu in Abhängigkeit von dem im ersten Parameter festgelegten Modus bestimmt, welcher einen der drei zuvor aufgezählten Datenbereiche beschreibt. Beim Einfügen beziehungsweise Ändern von Datenbankinhalten wird zudem eine Datenstruktur des Typs *TUserData* als Parameter an die jeweilige Methode übergeben, welche die einzufügenden beziehungsweise zu ändernden Werte für die betreffenden Spalten der jeweiligen Tabelle enthält.

4.3 Ressourcenverwaltung

Die Umsetzung der im *Kapitel 3.2* beschriebenen Konzeption zur Ermittlung und Auswertung der verfügbaren Systemressourcen des Client-Rechners ist Gegenstand dieses Kapitels. Während die Ressourcenermittlung auf der Client-Seite stattfindet, erfolgt die Auswertung dieser Daten durch den zentralen Server. Die Server-Module der einzelnen Zusatzdienste stellen dem zentralen Server hierzu eine Tabelle mit den von ihnen beanspruchten Mindestanforderungen an Client-Ressourcen zur Verfügung. Bevor hierzu jedoch nähere Einzelheiten dargelegt werden, folgt zunächst die Beschreibung der Implementierung der Ressourcenermittlung durch die Client-Anwendung.

Implementierung im Client-Modul

Zur Ermittlung der verfügbaren Ressourcen wird die in Microsoft Windows™integrierte Registry-Datenbank abgefragt. Hierzu wird unter anderem die zu diesem Zweck von [HOE02] entwickelte DLL in den zentralen Client eingebunden. Von den Methoden, welche durch diese DLL bereitgestellt werden, kommen die folgenden im vorliegenden System zum Einsatz:

- Methoden zur Ermittlung des Prozessortaktes
- Methoden zur Ermittlung der Größe des physischen Arbeitsspeichers.

Die Ermittlung der Bandbreite der aktuell verwendeten Netzanbindung erfolgt mit Hilfe einer frei verfügbaren Komponente namens *TMagRas*. Diese Komponente kann im Gegensatz zu der von [HOE02] entwickelten DLL auch die Bandbreite einer Modemverbindung ermitteln, was für die vorliegende Arbeit unbedingt notwendig ist.

Die von [HOE02] entwickelte DLL stellt auch eine Liste der Softwareprodukte bereit, welche auf dem jeweilig betrachteten System installiert sind. Diese Funktionalität ist aber zur Prüfung auf das Vorhandensein eines beliebigen Mail-Client nicht geeignet, da sie lediglich die Produktbezeichnungen beinhaltet, nicht aber deren Verwendungszweck. Zur Feststellung, ob ein für den integrierten eMail-Dienst benötigter Mail-Client auf dem Anwendersystem installiert ist, wird deshalb eine Funktion verwendet, die Bestandteil des zentralen Clients und im Rahmen dieser Arbeit entstanden ist. Diese greift ebenfalls auf die Registry-Datenbank zu und prüft dabei, ob ein Standard-Mail-Client im Anwendersystem registriert ist. Ist dies nicht der Fall, so ist auch kein Mail-Client auf dem System installiert. Der *Quelltext 4.16* zeigt die Implementierung dieser Abfrage.

```
function TfrmSvMain.CheckMailClientInstallation : Boolean;
var
  Reg: TRegistry;
  Key: String;
begin
  Result := False;
  Reg := TRegistry.Create;
  try
    Reg.RootKey := HKEY_CLASSES_ROOT;
    Key := 'mailto\shell\open\command';
    if Reg.OpenKey(Key,False) then Result := True;
  finally
    Reg.Free;
  end;
end;
```

Quelltext 4.16: Methode zur Prüfung des Vorhandenseins eines Mail-Clients

Zur Ermittlung, ob und welche Version des Internet Explorers™ auf dem Client-Rechner installiert ist, wird eine weitere frei verfügbare Komponente namens *TIEInstVers* verwendet. Diese ermittelt bei Vorhandensein einer solchen Software die Versionsnummer anhand der vom Internet Explorer verwendeten Bibliothek *SHDOCVW.DLL*, welche auch vom integrierten Web-Browser des zentralen Client verwendet wird.

Die Übermittlung aller hier ermittelten Ressourceninformationen an den zentralen Server erfolgt bei der Anmeldung des Nutzers an diesen. Hierzu wird die im *Quelltext 4.10* bereits beschriebene Datenstruktur verwendet. Ausgenommen hiervon ist die Information über das Vorhandensein eines Mail-Client, da der im zentralen Client integrierte eMail-Dienst entsprechend seiner Konzeption lediglich den Aufruf des Standard-Mail-Client beinhaltet und somit keine weiteren Ressourcenanforderungen berücksichtigen muss. Die Verfügbarkeit des Mail-Dienstes wird daher ohne die Auswertung dieser Ressourceninformation durch den zentralen Server allein von der Client-Anwendung übernommen.

Implementierung im Server-Modul

Die Mindestanforderungen für die verfügbaren Client-Ressourcen stellen der zentrale Dienst sowie die Server-Module der Zusatzdienste in Form der im *Quelltext 4.17* dargestellten Datenstruktur zur Verfügung. Der zentrale Dienst bietet hierbei zusätzlich noch die Information über die Mindestanforderung bezüglich der minimalen Versionsnummer des Internet Explorers™ an, da er den zur Präsentation der Vorlesungsinhalte integrierten Web-Browser beinhaltet.

```
TSvRequirementsRec = record
  ProcSpeed,          { Taktgeschwindigkeit in MHz}
  RAM     : Word;     { Arbeitsspeicher in MByte}
  Network : Cardinal ; {Bandbreite in Kbit/s}
end;
```

Quelltext 4.17: Der Record *TSvRequirementsRec*

Als Resultat der vorgenommenen Auswertung übermittelt der zentrale Server bei erfolgreicher Anmeldung eines Nutzers und Erfüllung der Ressourcen-bedingten Mindestanforderungen zur Nutzung des Präsentationsdienstes den im *Quelltext 4.11* abgebildeten Record an die Client-Anwendung, die diesen nach Erhalt auswertet und die Steuerelemente zur Nutzung der verfügbaren Dienste dementsprechend aktiviert oder sperrt.

4.4 Interaktionskomponenten

Die im *Kapitel 3.3* beschriebenen Konzepte für die im entworfenen System integrierten Interaktionskomponenten beinhalten unter anderem die Kommunikation zwischen Client- und Server-Modulen, die Abfrage und Auswertung von Systemressourcen sowie die Verwaltung von Gruppen und ihnen zugehörigen Nutzern. Nachdem die Beschreibung zur Umsetzung dieser Funktionalitäten in den *Kapiteln 4.1*, *4.2* und *4.3* erfolgt ist, werden hierauf aufbauend die Implementierungen für die einzelnen Dienste selbst vorgestellt. Begonnen wird hierbei mit dem Dienst zur Präsentation der Vorlesungsinhalte.

4.4.1 Dienst zur Präsentation der Vorlesungsinhalte

Der Dienst zur Präsentation der Vorlesungsinhalte stellt den Hauptanwendungszweck beim Distance-Learning dar und bildet somit gemeinsam mit dem Sitzungsmanager das Kernstück des hier entwickelten Systems. Deshalb beinhaltet die für diesen Dienst entwickelte Klasse *TfrmClMain* auch alle Routinen und Objekte, die zur Anmeldung eines Anwenders an das Distance-Learning-System und zur Darstellung der digital aufbereiteten Lehrinhalte erforderlich sind.

Zur Darstellung der angesprochenen Lehrinhalte wird laut Konzeption ein in die Client-Anwendung integrierter Web-Browser verwendet. Hierzu wird eine von Borland Delphi™ zur

Verfügung gestellte Komponente namens *TWebBrowser* eingesetzt, welche die Schnittstelle *IWebBrowser2* der Microsoft-Systembibliothek SHDOCVW.DLL kapselt [BOR01] und somit alle Funktionalitäten des auf dem Anwendersystem installierten Internet Explorers^{TM}innerhalb der Benutzeroberfläche der Client-Anwendung bereitstellt.

Zur Bereitstellung der Lehrinhalte in Form von HTML-Seiten wird ein unabhängig vom zentralen Server arbeitender Web-Server verwendet. Dieser kann hierbei sowohl auf demselben Rechnersystem wie der zentrale Server laufen als auch auf einem anderen. Der Vorteil dieses Umstandes besteht darin, dass eine eventuell hohe Auslastung der Netzanbindung des Web-Servers keinerlei Einfluss auf die Erreichbarkeit des zentralen Servers hat, falls beide Server auf verschiedenen Rechnersystemen mit separater Netzanbindung laufen.

Die in den hier beschriebenen HTML-Seiten integrierten Audio/Video-Daten werden von einem Streaming-Server bereitgestellt und innerhalb der HTML-Seite über ein entsprechend eingebundenes ActiveX-Steuerelement wiedergegeben. Bei der Implementierung dieser Funktionalität werden zunächst die folgenden drei Systeme betrachtet:

- **Windows Media**: Streaming-Technologie der Firma Microsoft^{TM}
- **RealMedia**: Streaming-Technologie der Firma RealNetworks^{TM}
- **QuickTime**: Streaming-Technologie der Firma Apple^{TM}.

In Hinsicht auf die Qualität der zum Zweck der Übertragung entsprechend komprimierten Audio- und Videosignale gibt es zwischen den hier betrachteten Systemen kaum Unterschiede. Auch bieten sie alle die Möglichkeit, die Wiedergabe der übertragenen Daten innerhalb einer HTML-Seite einzubinden. Ein wesentlicher Unterschied besteht aber in Hinblick auf die zur Verteilung der Medieninhalte angebotenen Streaming-Server. Der von Microsoft^{TM}angebotene *Windows Media Server^{TM}* wird als einziger dieser Server kostenlos zur Verfügung gestellt. Gleiches gilt auch für die sogenannten Encoder. Dabei handelt es sich um Werkzeuge, die im Rahmen der Aufbereitung von Audio/Video-Daten verwendet werden und dabei die originalen Videodaten in das vom jeweiligen System benötigte Format konvertieren [KUE01]. Der finanzielle Aspekt und die Tatsache, dass sich die drei hier betrachteten Systeme vom Funktionsumfang ähnlich sind, bilden letztendlich die ausschlaggebende Entscheidung dafür, im vorliegenden Distance-Learning-System *Windows Media^{TM}* als Technologie zur Übertragung und Darstellung der zur Lehrpräsentation eingesetzen Audio/Video-Inhalte zu verwenden.

Bei Verwendung von *Windows Media^{TM}* als Übertragungstechnologie wird der *Windows Media Player^{TM}* in der Version 7 zur Wiedergabe der Audio/Video-Inhalte verwendet. Dieser lässt sich in eine HTML-Seite als ActiveX-Steuerelement mit Hilfe des in HTML definierten OBJECT-Tags einbinden. Der nachfolgende *Quelltext 4.18* zeigt dies an einem Beispiel.

```
<HTML>
  <HEAD>
    <TITLE>Einbinden des Media Players 7 als ActiveX−Control in HTML−Seiten</TITLE>
```

```
</HEAD>
<BODY>
    <OBJECT
    ID="Player"
    CLASSID="CLSID:6bf52a52−394a−11d3−b153−00c04f79faa6"
    WIDTH=320
    HEIGHT=275
    TYPE="application/x−oleobject">
        <PARAM name="URL" value="mms://servername/filename.wmv">
    </OBJECT>
    <SCRIPT LANGUAGE="VBScript">
    <!−−
        On error resume next
        if err then
            msgbox " Sie benoetigen Windows Media Player 7 oder hoeher." & chr(13) &_
                "Download unter http :// www.microsoft.com/windowsmedia."
        err . clear
    −−>
    </SCRIPT>
</BODY>
</HTML>
```

Quelltext 4.18: Einbinden des Media Players als ActiveX-Control in HTML-Seiten

Die im *Quelltext 4.18* aufgeführten Attribute werden nachfolgend näher erläutert [KUE01, MUE02]:

- *ID*: Identifizierung des ActiveX-Controls innerhalb der Seite
- *CLASSID*: Referenzierung der Implementierung des gewünschten ActiveX-Controls (im Beispiel ist dies der Media Player 7)
- *WIDTH*: Breite des angezeigten Videofensters vom Media Player
- *HEIGHT*: Höhe des angezeigten Videofensters vom Media Player
- *TYPE*: Angabe, dass ein ActiveX-Control eingebunden wird
- *NAME*: Angabe des Parameternamens, welcher an den Media Player übergeben wird
- *VALUE*: Angabe der URL, über welche der Media Player auf die abzuspielende Datei zugreifen soll.

Ist das AxtiveX-Control des Media Players nicht auf dem Client-Rechner installiert, so wird die aus dem *Quelltext 4.18* ersichtliche Meldung ausgegeben.

Die Generierung der im *Kapitel 3.3.1* bereits beschriebenen Multistream-fähigen Videodateien erfolgt im vorliegenden System aufgrund der Auswahl von *Windows Media*^{TM} mit Hilfe des *Windows Media Encoders*^{TM}*9*. Dieses Werkzeug arbeitet unabhängig vom Distance-Learning-System und dient zur Aufbereitung der Original-Videodateien, die hierbei in das benötigte

Windows Media Format umgewandelt werden. Beim dabei ausgeführten Konvertierungsvorgang können alle Qualitätsstufen angegeben werden, welche die hierbei erzeugte Multistream-Videodatei zur Verfügung stellen soll.

Die Nutzung des Präsentationsdienstes durch einen Anwender setzt voraus, dass sich dieser mit einer gültigen Nutzerkennung und zugehörigem Passwort am zentralen Server anmeldet. Die dabei ablaufenden Vorgänge und zur Kommunikation verwendeten Datentypen werden bereits im *Kapitel 4.2* beschrieben und deshalb an dieser Stelle nicht nochmals dargestellt.

4.4.2 Chat

Der Chat-Dienst wird mit seinen Routinen und Objekten zum Datenaustausch sowie zur visuellen Darstellung der dabei anfallenden Informationen Client-seitig in der Klasse *TfrmChat* und Server-seitig in der Klasse *TfraChat* gekapselt. Neben der eigentlichen Textnachricht muss der Chat-Server auch die im *Kapitel 3.3.2* angesprochene ID des Verfassers an die zur Gruppe gehörigen Clients senden, womit sich die Frage stellt, wie diese Daten über die Socket-Verbindung übertragen werden können. Ein Record, welcher die ID und die Textnachricht enthält und mittels der Methode *SendBuf* des Socket-Objektes versendet wird, kommt in Hinblick auf ein geringes Datenaufkommen nicht in Frage, da in einem Record enthaltene Felder zur Aufnahme von Zeichenketten mit einer fester Länge definiert sein müssen. Mit anderen Worten bedeutet dies, dass ein solcher Record eine Gesamtgröße von 256 Byte aufweisen würde, da entsprechend der Konzeption von einem Nutzer Nachrichten mit einer Länge von bis zu 255 Zeichen versendet werden können und die Absender-ID ebenfalls 1 Byte Speicher beansprucht. Da ein Chat-Teilnehmer aber in der Regel wesentlich kürzere Nachrichten an die Gruppe senden wird, widerspricht ein solcher Record, der eben unabhängig von der tatsächlich zu übermittelnden Anzahl von Zeichen bei jeder zu versendenden Nachricht 256 Byte Übertragungsdaten verursachen würde, einem wichtigen Grundsatz der Konzeption, der ein minimales Datenaufkommen verlangt. Deshalb werden in der hier vorliegenden Implementierung die Nachricht eines Verfassers und dessen ID getrennt an die Chat-Clients versendet. Somit kann die Nachricht mittels der Socket-Methode *SendText* übermittelt werden, welche nur die der Länge der Zeichenkette entsprechende Anzahl an Bytes übermittelt. Das hat aber zur Bedingung, dass beide Seiten, also Client und Server, die Reihenfolge beim Versenden und Empfangen dieser beiden zusammengehörigen Informationen strikt beibehalten und alle Daten fehlerfrei übertragen werden. Diese Fehlerfreiheit wird durch das von den Sockets verwendete TCP-Protokoll garantiert. Nachfolgend sollen die zur Datenübertragung für diesen Dienst implementieren Routinen näher vorgestellt werden.

Routinen des Client-Moduls

In der Klasse *TfrmChat* sind unter anderem die innerhalb dieser Klasse globalen Variablen **ID** und **ReceiveID** definiert. Während ID zum Abspeichern der empfangenen Absender-ID verwendet wird, dient ReceiveID innerhalb des Chat-Clints zur Koordinierung des abwechselnden Empfangs von Absender-IDs und den zugehörigen Textnachrichten. Die Variable ReceiveID wird hierzu beim Eintritt in eine Chat-Gruppe mit dem Wert TRUE initialisiert, da der Chat-Server vor dem Senden einer Textnachricht an die Clients der zugehörigen Gruppe immer erst die ID des Verfasser überträgt.

Für die Zuordnung einer Textfarbe zum jeweiligen Nutzer wird für das vorliegende System eine Farbtabelle definiert. Dabei handelt es sich um ein Array, welches 16 verschiedene Farben über den Index 0 bis 15 bereitstellt. Die Auswahl des Index erfolgt hier durch Anwendung des Modulo-Operators auf die Absender-ID und den Wert 16, so dass immer ein Resultat aus dem Bereich von 0 bis 15 folgt.

Die Kennung des Absenders wird anhand der empfangenen Absender-ID und der dem Client zur Verfügung stehenden Liste mit Nutzerkennungen ermittelt. Die angesprochene Liste besitzt die Eigenschaft, dass die darin enthaltenen Nutzerkennungen zusammen mit der zugehörigen ID abgespeichert sind, welche dem Chat-Mitglied beim Eintritt in eine Gruppe vom Sitzungsmanager laut Konzeption zugewiesen wird. Mittels der zum Listenobjekt gehörenden Funktion *IndexOfObject* erhält man den Index des Listenelementes, welches die im Funktionsparameter angegebene ID und damit auch die gesuchte Nutzerkennung enthält.

Im *Quelltext 4.19* wird die Behandlungsroutine für das *OnRead*-Ereignis des vom Chat-Client verwendeten TClientSocket-Objektes dargestellt, welches die hier beschriebenen Vorgänge beinhaltet.

```
procedure TfrmChat.tcpClSockChatRead(Sender: TObject; Socket: TCustomWinSocket);
begin
  {ID des Verfassers aus Socket−Verbindung lesen}
  if (ReceiveID) then begin
    Socket.ReceiveBuf(ID,SizeOf(ID));
    ReceiveID := False;
  end
  { Textnachricht aus Socket−Verbindung lesen und gemeinsam mit Absenderkennung darstellen}
  else begin
    redBoard. SelAttributes .Color := ColorTable[ID mod 16]; {Festlegung der Textfarbe}
    with lstMember.Items do
      redBoard.Lines.Add(Format('%s:␣%s',[Strings[IndexOfObject(TObject(ID))], Socket.ReceiveText ]));
    ReceiveID := True;
  end;
end;
```

Quelltext 4.19: Empfang von Nachrichten beim Chat-Client

Das Versenden von Textnachrichten durch den Chat-Client erfolgt durch Betätigung der

hierfür vorgesehenen Schaltfläche beziehungsweise durch Drücken der ENTER-Taste zum Abschluss der Texteingabe. Dabei wird die im *Quelltext 4.20* dargestellte Routine ausgeführt, welche die zu versendende Nachricht an die Methode *SendText* des Socket-Objektes übergibt und anschließend das Eingabefeld leert.

```
procedure TfrmChat.btnChatSendClick(Sender: TObject);
begin
  if (Length(Trim(redInput.Text)) > 0) then with redInput do
  begin
    btnChatSend.Enabled := False;
    tcpClSockChat.Socket.SendText(Text);
    Clear;
  end;
end;
```

Quelltext 4.20: Versenden von Nachrichten durch den Chat-Client

Routinen des Server-Moduls

Im Modul des Chat-Servers, welches durch die Klasse *TfraChat* repräsentiert wird, erfolgt die Verwaltung der Gruppen und Nutzer dieses Dienstes sowie die Verteilung der Chat-Nachrichten an die Mitglieder der jeweiligen Gruppen. Während die bei der Nutzer- und Gruppenverwaltung auftretenden Programmabläufe und die dabei verwendeten Klassen und Methoden bereits im *Kapitel 4.2* in ihren Grundzügen erläutert wurden *(vgl. auch Quelltext 4.3)*, folgt an dieser Stelle eine Darstellung zur Umsetzung der Nachrichtenverteilung.

Hierzu wird die Ereignisroutine *OnClientRead* des im Modul des Chat-Servers enthaltenen TServerSocket-Objektes betrachtet, welche beim Eintreffen von Daten seitens der Chat-Clients die Verarbeitung dieser Nachrichten übernimmt. Dabei wird zunächst anhand der Eigenschaft *Data* des Socket-Objektes das zum Absender der Daten gehörige und vom Sitzungsmanager verwaltete Nutzerobjekt ermittelt. Danach erfolgt das Auslesen der Nachricht mittels der Methode *ReceiveText* aus der Socket-Verbindung. Hierauf wird unter Verwendung des Nutzerobjektes die entsprechende Chat-Gruppe ermittelt und anhand der darin enthaltenen Mitgliederliste der Versand der Absender-ID und Textnachricht für die zur Gruppe gehörenden Clients durchgeführt. Der nachfolgende *Quelltext 4.21* stellt diese Vorgänge nochmals dar.

```
procedure TfraChat.tcpSvSockChatClientRead(Sender: TObject; Socket: TCustomWinSocket);
var
  ...
begin
  User := TUser(Socket.Data);
  ...
  else begin {Chat-Nachricht empfangen und weiterleiten}
    ChatMsg := Socket.ReceiveText;
    {Nachricht an alle Mitglieder der Gruppe weiterleiten}
```

```
      for i:=User.ChatGroup.CountUser−1 downto 0 do begin
        {Senden der innerhalb der Chat−Gruppe gueltigen User−ID des Verfassers}
        User.ChatGroup.User[i].ChatSocket.SendBuf(User.ChatID,SizeOf(User.ChatID));
        {Senden der Nachricht}
        User.ChatGroup.User[i].ChatSocket.SendText(ChatMsg);
      end;
    end;
  end;
```

Quelltext 4.21: Empfang und Versand von Nachrichten beim Chat-Server

4.4.3 Audio- und Videokommunikation

Zur Implementierung der Nutzerkommunikation mittels Audio/Video-Konferenz wird im System GWOTS eine Sammlung von Bibliotheken verwendet, welche von [OPE03] entwickelt wurde und eine voll funktionsfähige Open Source-Implementierung des H.323-Protokolls beinhaltet. Die hier angesprochenen und in der Programmiersprache C++ implementierten DLL-Bibliotheken umfassen auch Objekte, so dass sie nicht direkt in die im vorliegenden System verwendete Delphi-Umgebung eingebunden werden können. Aus diesem Grund erfolgt das Einbinden dieser Bibliotheken über ein ActiveX-Steuerelement, welches im Rahmen dieser Arbeit mit Hilfe der Programmierumgebung *Microsoft Visual C++*^{TM} entwickelt wurde und alle zur Steuerung der Funktionalitäten erforderlichen Methoden und Eigenschaften bereitstellt.

Zur direkten Kommunikation zwischen zwei Nutzern wird seitens des Anwenders, der einen anderen Teilnehmer zu einer privaten Konferenz einladen möchte, die IP-Adresse des Zielrechners benötigt. Diese erfragt die Client-Anwendung vom zentralen Server, wenn sie diesem die Absicht des Konferenzaufbaus übermittelt. Der zentrale Server übermittelt hierauf bei Verfügbarkeit des Konferenzdienstes für den Zielrechner die gesuchte IP-Adresse in Form einer Zeichenkette mittels der Socket-Methode *SendText*, ansonsten wird eine Zeichenkette mit dem Inhalt ’0’ gesendet und dementsprechend vom Client ausgewertet.

Beim Eintritt in eine Konferenzgruppe beziehungsweise der Eröffnung einer solchigen wird vom Dienst-Client-Modul über die vom bereits angesprochenen ActiveX-Steuerelement zur Verfügung gestellten Methode *Connect* eine Verbindung zum Konferenz-Server hergestellt, bei welcher der Gruppenname als Parameter in der Form *group_name@server_ip* übergeben wird. Über den Steuer-Socket des zentralen Client wird dem zentralen Server zum Zweck des einheitlichen Sitzungsmanagements lediglich dieser Verbindungsaufbau inklusive des Gruppennamens mitgeteilt.

Wird eine bestehende Konferenz von einem Nutzer verlassen, so wird dem zentralen Server eine entsprechende Meldung übermittelt, um die zum Zweck des einheitlichen Sitzungsmanagements geführten Gruppenlisten aktualisieren zu können. Zur Übermittlung dieser Informationen bezüglich des Auf- oder Abbaus einer privaten beziehungsweise Gruppenkonferenz wird der im *Kapitel 4.2* zur Listenaktualisierung vorgestellte Record *TListRec* verwendet.

Bei dem im GWOTS verwendeten Videokonferenz-Server (MCU) handelt es sich um eine eigenständig lauffähige Anwendung, welche unabhängig vom zentralen Server arbeitet und nicht im Rahmen dieser Arbeit entwickelt wurde. Dieser Server kann auf einem vom zentralen Server getrennten Rechnersystem eingesetzt werden, was analog zum verwendeten Web-Server und Streaming-Server bei Nutzung dieser Eigenschaft zur Entlastung der Netzanbindung des zentralen Servers beiträgt.

Die Bereitstellung und Aktualisierung der Listen mit den verfügbaren Konferenzgruppen und deren Teilnehmern erfolgt mit Hilfe der im *Kapitel 4.2* beschriebenen Datentypen und Aktionen.

4.4.4 Whiteboard

Analog zum Chat-Dienst erfolgt auch beim Whiteboard-Dienst eine Kapselung der Routinen und Objekte für das Dienst-Server-Modul und Dienst-Client-Modul in den entsprechend dafür generierten Klassen namens *TfraWb* und *TfrmWb*. Die Anmeldung eines Anwenders an eine Whiteboard-Gruppe beinhaltet hierbei die Abläufe, wie sie im *Kapitel 4.2* beschrieben werden.

Zur Ausführung von Zeichenoperationen wird die Methode *Draw* definiert, welcher ein Record namens *TDrawAction* als Parameter übergeben wird und die anhand der Record-Informationen die entsprechenden Aktionen zur grafischen Ausgabe ausführt. In dieser Record-Struktur sind alle laut Konzeption erforderlichen und im *Kapitel 3.3.4* aufgeführten Parameter enthalten, die zur Beschreibung der hier möglichen Zeichenoperationen erforderlich sind. Der nachfolgende *Quelltext 4.22* zeigt diese Struktur sowie den darin enthaltenen Datentyp, welcher die Art der mit dem jeweiligen Record beschriebenen Zeichenoperation kennzeichnet.

```
TToolType = ( ttPolygon , ttLine , ttCircle , ttRect , ttRndRect , ttPipette , ttFill , ttErazor , ttText );

TDrawAction = record
  x1,y1,
  x2,y2   : Integer ;          {Koordinaten}
  Tool    : TToolType;         {Art der Zeichenoperation}
  Pen     : TPen;              {Zeiger auf TPen-Objekt}
  Brush   : TBrush;            {Zeiger auf TBrush-Objekt}
  Font    : TFont;             {Zeiger auf TFont-Objekt}
  Text    : String [255];      {Zeichenkette beim Einfuegen von Text}
end;
```

Quelltext 4.22: Der Record *TDrawAction*

Die hierin enthaltenen Objekte des Typs *TPen*, *TBrush* und *TFont* gehören zum Steuerelement des Typs *TImage*, welches Bestandteil der Delphi-VCL ist und im hier vorliegenden System zur Ausgabe der grafischen Informationen, also der Zeichenoperationen, verwendet wird. Der Verwendungszweck dieser drei Objekte wird nachfolgend kurz erläutert:

- *TPen* beschreibt die Attribute eines Windows-Stiftes, der ein Bild auf einer Zeichenfläche darstellt [BOR01]. Von besonderem Interesse sind hierbei die Art, Stärke und Farbe der zu zeichnenden Linie.

- *TBrush* kapselt das Windows-Pinselobjekt (HBRUSH) und wird zum Füllen geschlossener Flächen (z.B. Rechtecke und Ellipsen) mit einer Farbe oder einem Muster verwendet [BOR01]. Von besonderem Interesse sind hierbei das Füllmuster und die Füllfarbe.

- *TFont* beschreibt die Schriftmerkmale für die Anzeige von Text [BOR01]. Im vorliegenden System sind hierbei die Größe und Farbe der Schrift von besonderem Interesse.

Da der Record *TDrawAction* lediglich die Zeiger auf die drei zuvor näher beschriebenen Objekte enthält, und diese nur lokal Gültigkeit besitzen, müssen zur Fernübertragung der Zeichenoperationen anstatt dieser Zeigerobjekte die tatsächlichen Werte übermittelt werden, welche in der vorangegangenen Aufzählung für jedes der drei Objekte angegeben sind. Außerdem sieht das Konzept zum Whiteboard-Dienst laut *Kapitel 3.3.4* eine Komprimierung der zu übermittelnden Daten vor, um das Datenaufkommen beim Whiteboard-Server gering zu halten. Die *Abbildung 3.8* beschreibt hierbei die Datenstruktur, wie sie das Konzept vorschreibt.

Zur Implementierung dieser Struktur wird der 8 Byte umfassende Datentyp *TWbDrawRec* definiert. Die Speicherung der Informationen erfolgt hierbei in der Reihenfolge, wie sie die *Abbildung 3.8* vorgibt. Hierzu werden die von Delphi bereitgestellten logischen Operatoren **shl** und **or** verwendet, welche eine Bitverschiebung nach links beziehungsweise eine bitweise ODER-Operation realisieren. Die zum Zweck einer solchen Datenkomprimierung im GWOTS definierte Funktion *CompressWbDrawRec* liefert als Rückgabewert den beschriebenen Record vom Typ *TWbDrawRec*, welcher von dieser Funktion anhand des als Parameter übergebenen TDrawAction-Objektes mit den entsprechenden Werten belegt wird. Der *Quelltext 4.23* zeigt den Ablauf der dabei ausgeführten Aktionen und die Verwendung der bereits angesprochenen Operatoren.

```
function CompressWbDrawRec(ADrawAction: TDrawAction): TWbDrawRec;
begin
  with ADrawAction do
  begin
    {Werkzeug}
    Result := Byte(Tool);
    {Koordinaten x1, y1, x2 und y2}
    Result := Result shl 10 or x1;
    Result := Result shl 10 or y1;
    Result := Result shl 10 or x2;
    Result := Result shl 10 or y2;
    {Linienart}
    Result := Result shl 3 or Byte(Pen.Style);
    {Linienfarbe / Schriftfarbe }
    Result := Result shl 4 or ColorToIndex(Pen.Color);
```

```
    { Linienstaerke / Schriftgroesse }
    Result := Result shl 6 or Pen.Width;
    {Fuellmuster}
    Result := Result shl 3 or Byte(Brush.Style );
    { Fuellfarbe }
    Result := Result shl 4 or ColorToIndex(Brush.Color );
  end;
end;
```

Quelltext 4.23: Funktion zur Komprimierung der Whiteboard-Daten

Um die auf diese Art kodierten und per Datenversand übermittelten Daten wieder in einen Record vom Typ *TDrawAction* umzuwandeln, der bekanntlich zur Ausführung einer Zeichenoperation an die Methode *Draw* übergeben wird, beinhaltet die im GWOTS definierte Methode *DecompressWbDrawRec* alle Aktionen, die hierzu notwendig sind. Dabei wird neben dem TWbDrawRec-Objekt auch die Adresse eines TDrawAction-Objektes als Parameter übergeben, welches die aus dem TWbDrawRec-Objekt extrahierten Informationen zur weiteren Verwendung aufnimmt. Zum Extrahieren der Daten werden die logischen Operatoren **shr** und **and** verwendet, welche eine Bitverschiebung nach rechts beziehungsweise eine bitweise UND-Operation realisieren. Der UND-Operator dient hierbei in Verbindung mit fest definierten Konstanten zur Maskierung der im Datentyp *TWbDrawRec* enthaltenen Bitfolge. Die Anwendung der hier beschriebenen Oparatoren sowie die Definition der verwendeten Konstanten sind aus dem *Quelltext 4.24* ersichtlich, welcher die zur Dekomprimierung der übermittelten Whiteboard-Daten definierte Methode *DecompressWbDrawRec* darstellt.

```
procedure DecompressWbDrawRec(AWbDrawRec: TWbDrawRec; var ADrawAction: TDrawAction);
const
  _10bit = $03FF; _6bit  = $003F; _4bit  = $000F; _3bit  = $0007;
begin
  with ADrawAction do
  begin
    Brush.Color := ColorTable[AWbDrawRec and _4bit];  { Fuellfarbe }
    AWbDrawRec := AWbDrawRec shr 4;
    Brush.Style := TBrushStyle(AWbDrawRec and _3bit); {Fuellmuster}
    AWbDrawRec := AWbDrawRec shr 3;
    Pen.Width := AWbDrawRec and _6bit;                { Liniestaerke / Schriftgroesse }
    AWbDrawRec := AWbDrawRec shr 6;
    Pen.Color := ColorTable[AWbDrawRec and _4bit];    { Linienfarbe / Schriftfarbe }
    AWbDrawRec := AWbDrawRec shr 4;
    Pen.Style := TPenStyle(AWbDrawRec and _3bit);     { Linienart }
    AWbDrawRec := AWbDrawRec shr 3;
    y2 := AWbDrawRec and _10bit;                      {Koordinaten y2, x2, y1 und x1}
    AWbDrawRec := AWbDrawRec shr 10;
    x2 := AWbDrawRec and _10bit;
    AWbDrawRec := AWbDrawRec shr 10;
    y1 := AWbDrawRec and _10bit;
    AWbDrawRec := AWbDrawRec shr 10;
    x1 := AWbDrawRec and _10bit;
```

```
      AWbDrawRec := AWbDrawRec shr 10;
      Tool  :=  TToolType(AWbDrawRec and ¸4bit);          {Werkzeug}
    end;
  end;
```

Quelltext 4.24: Methode zur Dekomprimierung der Whiteboard-Daten

Nach der hier dokumentierten Beschreibung der beim Whiteboard-Dienst eingesetzten Datentypen und den damit verbundenen Methoden wird nachfolgend eine kurze Übersicht über die Programmabläufe im Client-Modul beziehungsweise Server-Modul des hier beschriebenen Dienstes vermittelt.

Programmabläufe im Client-Modul

Die Ausführung einer Zeichenoperation durch die Methode *Draw* sowie der Versand der mit dem dabei verwendeten TDrawAction-Objekt korrespondierenden Daten erfolgt, nachdem der Anwender die betreffende Operation per Maus auf der ihm durch den Whiteboard-Client zur Verfügung gestellten Zeichenfläche durchgeführt hat und dabei den Abschluss der Zeichenoperation durch das Loslassen der Maustaste kenntlich macht. Die Zeichnungsinformationen werden dabei vor dem Versenden an das Dienst-Server-Modul entsprechend der vorangegangenen Ausführungen komprimiert. Der Versand selbst erfolgt mittels der Methode *SendBuf* des verwendeten Socket-Objektes. Betrifft die ausgeführte Zeichenoperation das Einfügen von Text, so wird die hierzu gehörige Zeichenkette unmittelbar nach dem Versenden des TWbDrawRec-Objektes mit Hilfe der Socket-Methode *SendText* an den Server verschickt.

Während beim Beginn der Zeichenoperation, also beim Niederdrücken der Maustaste innerhalb des Zeichenfläche, die Ausgangskoordinaten und die Art der Zeichenoperation im dafür vorhandenen TDrawAction-Objekt vermerkt werden, erfolgt beim Loslassen der Maustaste die Zuordnung der Endkoordinaten entsprechend der aktuellen Mausposition. Die Einstellungen bezüglich der beschriebenen Eigenschaften für den Zeichenstift, das Füllmuster beziehungsweise die Schrift erfolgt über entsprechende Menüs vor Ausführung einer Zeichenoperation. Diese Einstellungen behalten dann so lange ihre Gültigkeit für alle darauffolgenden Zeichenoperationen, bis sie vom Anwender geändert werden.

Werden Daten vom Server empfangen und entsprechend über die Socket-Verbindung mittels der Methode *ReceiveBuf* des Socket-Objektes gelesen, so erfolgt hierauf die Dekomprimierung dieser Daten und deren grafische Darstellung auf der zur Verfügung stehenden Zeichenfläche des Whiteboard-Clients. Bezeichnet die Art der Zeichenoperation dabei das Einfügen von Text, so wird hier vor dem Aufruf der Methode *Draw* zusätzlich mittels der Methode *ReceiveText* des Socket-Objektes die zughörige Zeichenfolge aus der Socket-Verbindung ausgelesen und der Eigenschaft *Text* des TDrawAction-Objektes zugewiesen.

Programmabläufe im Server-Modul

Das Server-Modul des Whiteboard-Dienstes hat neben der Weiterverteilung von empfangenen Zeichenoperationen auch die Aufgabe, die Zeichenoperationen für jede existierende Gruppe getrennt in dafür vorhandenen Listen abzuspeichern. Dies begründet sich aus dem hierzu beschriebenen Konzept, welches beinhaltet, dass einem Nutzer beim Eintreten in eine bereits bestehende Gruppe vom Server der aktuelle Arbeitsstand dieser Gruppe übermittelt wird. Die hier angesprochenen Listen werden von der im *Kapitel 4.2* vorgestellten Klasse *TGroup* bereitgestellt und innerhalb des Gruppenobjektes mittels der Eigenschaften *WbDrawRecList* beziehungsweise *WbTextRecList* angesprochen.

Beim Empfang von Daten wird innerhalb der *OnClientRead*-Behandlungsroutine des TServerSocket-Objektes die Socket-Methode *ReceiveBuf* verwendet, um die TWbDrawRec-Objekte zu empfangen. Diese werden daraufhin in der entsprechenden Liste des Gruppenobjektes abgespeichert und an die Mitglieder der zum Absender gehörigen Gruppe weitergeleitet. Sollte die empfangene Zeichenoperation das Einfügen von Text beschreiben, so wird unmittelbar nach Empfang des TWbDrawRec-Objektes die Socket-Methode *ReceiveText* verwendet, um die der Zeichenoperation zugehörige Zeichenfolge aus der Socket-Verbindung zu lesen. Diese empfangene Zeichenfolge wird dann ebenfalls in der dafür vorhandenen Liste des Gruppenobjektes vermerkt und an die Gruppenmitglieder weitergeleitet.

Beim Eintritt eines Nutzer in eine Gruppe werden diesem alle in der vom Gruppenobjekt bereitgestellten Liste verfügbaren Zeichenoperationen übermittelt. Dabei prüft das Dienst-Server-Modul, ob die akzuell übermittelte Zeichenoperation das Einfügen von Text beschreibt. Sollte dies der Fall sein, so wird als nächste Aktion die korrespondierende Zeichenkette aus der entsprechenden Liste des Gruppenobjektes gesendet, bevor mit dem Versand der restlichen Zeichenoperationen fortgefahren wird.

4.4.5 FTP und eMail

Die Zusatzdienste *FTP* und *eMail* sind laut Konzeption kein fester Bestandteil der zentralen Clients. Es handelt sich bei diesen Diensten um eigenständige Anwendungen, die lediglich aus der Benutzeroberfläche der Client-Anwendung heraus gestartet und dabei nicht in das einheitliche Sitzungsmanagement einbezogen werden. Zu diesem Zweck werden im Client-Modul entsprechende Startroutinen implementiert, welche die geforderte Funktionalität umsetzen. Hierzu wird zum Aufruf des FTP-Dienstes die Funktion *ShellExecute* der Windows-API verwendet, welche das Starten externer Programme aus einer Anwendung heraus erlaubt. Der folgende *Quelltext 4.25* zeigt die Anwendung dieser Funktion für den Start des FTP-Dienstes.

```
procedure TfrmClMain.btnFtpClick(Sender: TObject);
begin
  ShellExecute(Handle,'open',PChar('Ftp.exe'),PChar(''),PChar(GetCurrentDir),SW_SHOWNORMAL);
```

```
end;
```

Quelltext 4.25: Methode zum Aufruf des FTP-Clients

Die an die Funktion *ShellExecute* zu übergebenden Parameter besitzen dabei folgende Bedeutung [BOR01]:

- *1. Parameter*: Dieser Parameter beinhaltet das Handle des aufrufenden Programmfensters.
- *2. Parameter*: Im 2. Parameter wird die auszuführende Operation angegeben. Im vorliegenden Fall muss hierfür 'open' angegeben werden, da eine Datei ausgeführt werden soll.
- *3. Parameter*: Der Dateiname des auszuführenden Programms ist im 3. Parameter vermerkt.
- *4. Parameter*: Der 4. Parameter bietet die Möglichkeit zur Angabe von Parametern, welche der ausführbaren Datei bei deren Aufruf übergeben werden.
- *5. Parameter*: Das Arbeitsverzeichnis wird im 5. Parameter angeben. Im hier vorliegenden Beispiel ist es das aktuelle Verzeichnis.
- *6. Parameter*: Der letzte Parameter gibt an, wie das aufgerufene Programm angezeigt werden soll. Möglichkeiten sind hier beispielsweise minimiert oder maximiert. Im vorliegenden Fall wird das Programmfenster des FTP-Clients in seiner normalen Größe angezeigt.

Der Aufruf des eMail-Dienstes hingegen erfolgt mit Hilfe des im zentralen Client integrierten Web-Browser-Objektes. Diesem wird als zu öffnende URL der Parameter *'mailto:'* übergeben, woraufhin das Browser-Objekt den auf dem Anwendersystem installierten Standard-Mail-Client aufruft. Der *Quelltext 4.26* stellt diese Implementierung dar.

```
procedure TfrmClMain.btnMailClick(Sender: TObject);
begin
  brwBrowser.Navigate('mailto :' );
end;
```

Quelltext 4.26: Methode zum Aufruf des eMail-Clients

4.4.6 Kurznachrichten

Der Datenversand des Dienstes zur Übermittlung von Kurznachrichten wird über den Steuer-Socket des zentralen Servers beziehungsweise Clients realisiert. Dem Anwender steht zur Nutzung dieses Dienstes eine Liste mit den Kennungen aller am System angemeldeten Nutzer zur

Verfügung, deren Bereitstellung und Aktualisierung bereits im *Kapitel 4.2* beschrieben wird. Die Client-Anwendung versendet hierzu einen Record vom Typ *TMsgRec*, welcher neben der Nachricht und der Absenderkennung auch die Adresse des zum Empfänger gehörigen Nutzerobjektes enthält. Durch Auswertung dieser Adresse erhält der Server das Steuer-Socket-Objekt des Empfängers der Nachricht und leitet diese entsprechend weiter. Zum Senden und Empfangen des Records *TMsgRec*, dessen Struktur im *Quelltext 4.27* dargestellt wird, erfolgt sowohl Server-seitig als auch Client-seitig mittels der Socket-Methoden *SendBuf* beziehungsweise *ReceiveBuf*.

```
TMsgRec = record
  Sender   : String [20];
  Receiver : Integer;
  Msg      : String [255];
end;
```

Quelltext 4.27: Der Record *TMsgRec*

Eine Optimierung bezüglich des Datenumfangs der zu übermittelnden Nachricht, wie dies beispielsweise beim Chat-Dienst betrieben wird, ist beim Dienst für Kurznachrichten nicht erforderlich, da die durch diesen Dienst übermittelten Nachrichten vom Server nicht an eine Gruppe verteilt werden müssen, sondern lediglich an einen einzigen Empfänger.

4.5 Nutzung des Systems

Im Folgenden wird in kurzer Form eine Beschreibung zur Nutzung des hier entwickelten Distance-Learning-Systems gegeben. Hierzu werden die grafischen Oberflächen und ihre Funktionalität sowohl für das Server-Modul als auch die Client-Anwendung in ihren Grundzügen vorgestellt.

Server-Modul

Nach dem Start der Server-Anwendung befindet sich diese zunächst im OFFLINE-Modus. Um den Server für die Client-Anwendungen verfügbar zu machen, muss dieser vom Administrator gestartet werden. Die dabei erforderliche Vorgehensweise sowie die Erklärung weiterer Funktionalitäten werden anhand der aus *Abbildung 4.2* ersichtlichen Steuerelemente erklärt, wobei diese Abbildung einen Screenshot des hier entwickelten Server-Moduls zeigt.

Die aus der *Abbildung 4.2* ersichtliche Menüleiste und Werkzeugleiste des Server-Moduls bieten dabei die nachfolgend beschriebenen Funktionalitäten:

- Menüpunkt **Server**

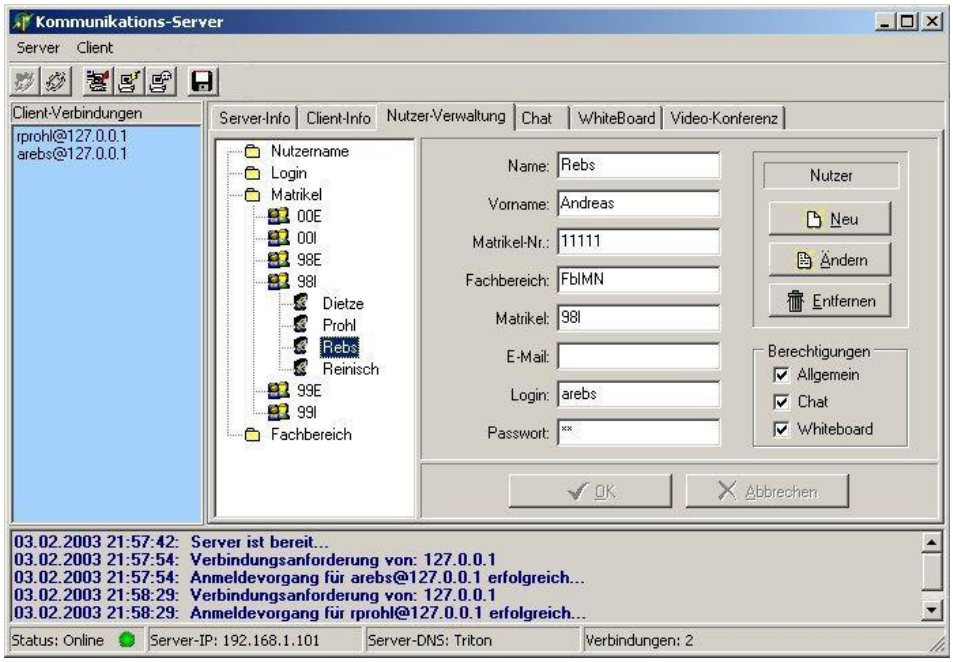

Abbildung 4.2: Nutzerverwaltung im Server-Modul

- Die Menüoption **Starten** versetzt das Server-Modul in den ONLINE-Modus, so dass Client-Anforderungen vom zentralen Server und den Zusatzdient-Servern entgegengenommen und bearbeitet werden können. Analog zur hier beschriebenen Menüoption kann auch der erste Tool-Button der Werkzeugleiste verwendet werden.
- Durch Auswahl der Menüoption **Herunterfahren** wird das Server-Modul in den OFFLINE-Modus versetzt. In diesem Modus sind der zentrale Server und die Dienst-Server-Module nicht für Client-Anforderungen erreichbar. Ein Herunterfahren des Servers ist aber nur möglich, wenn zu diesem Zeitpunkt keine Clients am Server angemeldet sind. Analog zur hier beschriebenen Menüoption kann auch der zweite Tool-Button der Werkzeugleiste verwendet werden.
- Alle beim Server-Modul auftretenden Vorgänge, wozu neben dem Starten und Herunterfahren des Servers auch alle eingehenden Client-Anforderungen für sämtliche Dienste gehören, werden protokolliert und können durch Auswahl der Menüoption **Log-Datei speichern** in einer Textdatei abgespeichert werden. Analog zur hier beschriebenen Menüoption kann auch der sechste Tool-Button der Werkzeugleiste verwendet werden.
- Die Menüoption **Beenden** dient zum Schließen des Server-Moduls. Hierbei ist zu beachten, dass das Server-Modul nur geschlossen werden kann, wenn der Server

heruntergefahren ist. Ansonsten wird eine entsprechende Meldung ausgegeben, die hierauf hinweist.

- Menüpunkt **Client**
  - Bei Auswahl der Menüoption **Alle zurücksetzen** kann der Administrator die Verbindungen für alle am System angemeldeten Nutzer beenden. Hierbei wird eine Eingabebox geöffnet, in welche eine Begründung für den Server-seitigen Verbindungsabbau eingegeben werden kann. Diese Begründung wird auf Client-Seite angezeigt, bevor der Verbindungsabbau erfolgt. Analog zur hier beschriebenen Menüoption kann auch der dritte Tool-Button der Werkzeugleiste verwendet werden.
  - Die Menüoption **Zurücksetzen** funktioniert analog zur Menüoption *Alle zurücksetzen*, nur dass hier nicht alle Clients zum Verbindungsabbau aufgefordert werden, sondern nur ein Einzelner. Dieser wird aus der in *Abbildung 4.2* auf der linken Seite des Programmfensters ersichtlichen Liste ausgewählt. Analog zur hier beschriebenen Menüoption kann auch der vierte Tool-Button der Werkzeugleiste verwendet werden.
  - Die Menüoption **Nachricht senden** erlaubt es dem Administrator, eine Textnachricht an einen am System angemeldeten Nutzer zu verschicken. Hierbei wird eine Eingabebox geöffnet, in welche die zu versendende Nachricht eingegeben wird. Die Auswahl des Empfängers erfolgt anhand der Liste, welche in *Abbildung 4.2* auf der linken Seite des Programmfensters zu sehen ist. Analog zur hier beschriebenen Menüoption kann auch der fünfte Tool-Button der Werkzeugleiste verwendet werden.

Wie aus *Abbildung 4.2* außerdem ersichtlich wird, gliedert sich die Benutzeroberfläche in drei Bereiche. Während im linken Bereich alle angemeldeten Nutzer inklusive der IP-Adresse des von ihnen verwendeten Rechners in der Form *nutzername@ip_adresse* aufgelistet werden, erfolgt im unteren Bereich die Anzeige aller beim Server auftretenden Ereignisse. Hierzu gehören neben den An- und Abmeldevorgänge von Nutzern und eventuell auftretenden Systemereignissen auch die Inhalte des Dienstes für Kurznachrichten.

Der Hauptbereich der Benutzeroberfläche bietet über hierfür verfügbare Registerkarten Zugriff auf die grafischen Steuerelemente der einzelnen Dienst-Server-Module sowie der Datenverwaltung. Das in *Abbildung 4.2* dargestellte Nutzerinterface zur Datenverwaltung gliedert sich dabei seinerseits in zwei Bereiche. Während im linken Bereich die Nutzernamen, Logins, Matrikel und Fachbereiche nach der im *Kapitel 3.1.2* beschriebenen hierarchischen Struktur angezeigt werden und zur Anzeige weiterer Details beziehungsweise zur Bearbeitung selektiert werden können, dient der rechte Bereich zur Ausgabe und Bearbeitung der ausgewählten Daten. Zum Anlegen neuer beziehungsweise Ändern oder Löschen bestehender Daten stehen

die aus *Abbildung 4.2* ersichtlichen Schaltflächen zur Verfügung, welche die entsprechenden Operationen einleiten.

Der in *Abbildung 4.3* dargestellte Screenshot zeigt ebenfalls das hier beschriebene Server-Modul, nur dass hier im Hauptbereich der Benutzeroberfläche die grafischen Steuerelemente des Chat-Servers abgebildet sind. Dieser Hauptbereich gliedert sich beim Chat-Server wiederum in zwei Bereiche, wobei im linken Bereich die am Chat-Server angemeldeten Gruppen und deren zugehörige Nutzer in Listenform angezeigt werden. Die Anzeige der Gruppenmitglieder erfolgt dabei in Abhängigkeit vom selektierten Eintrag der Gruppenliste. Im dargestellten Screenshot der *Abbildung 4.3* ist in diesem Fall eine Chat-Gruppe mit der Bezeichnung *MSAccess* ausgewählt, welcher die Mitglieder mit den Nutzerkennungen *rprohl* und *arebs* angehören. Im rechten Bereich des Hauptbereiches werden beim Chat-Server die von diesem empfangenen und weitergeleiteten Chat-Nachrichten angezeigt.

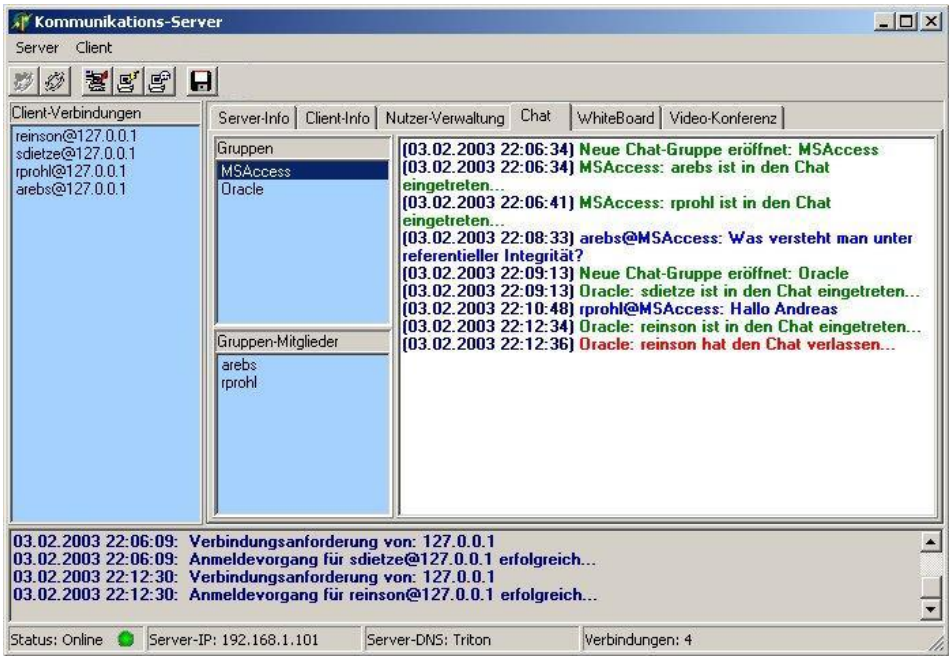

Abbildung 4.3: Chat-Dienst im Server-Modul

Die Anzeige der angemeldeten Gruppen und deren zugehöriger Nutzer sowie die Aufteilung des Hauptbereiches erfolgt für die Zusatzdienste Whiteboard und Viedeokonferenz analog zum hier beschriebenen Chat-Dienst. Während beim Whiteboard im rechten Bereich entsprechend der selektierten Whiteboard-Gruppe deren aktuell bearbeitetes Dokument dargestellt wird, werden beim Videokonferenz-Dienst an dieser Stelle keine Informationen angezeigt, da der Videokonferenz-Server nicht im zentralen Server integriert ist. Beim Videokonferenz-Dienst hat der Administrator also einen Überblick über die bestehenden Konferenzgruppen und deren Teilnehmer, nicht jedoch über die Konferenzinhalte.

Client-Modul

Analog zum Server-Modul befindet sich auch die Client-Anwendung nach dem Start zunächst im OFFLINE-Modus. Um auf die Server-Dienste zugreifen zu können, muss sich der Anwender mit Login und Passwort am zentralen Server anmelden. Die dabei erforderliche Vorgehensweise sowie die Erklärung weiterer Funktionalitäten werden anhand des in *Abbildung 4.4* dargestellten Screenshots beschrieben, welcher die Benutzeroberfläche des hier entworfenen Client-Moduls und dessen Steuerelemente zeigt.

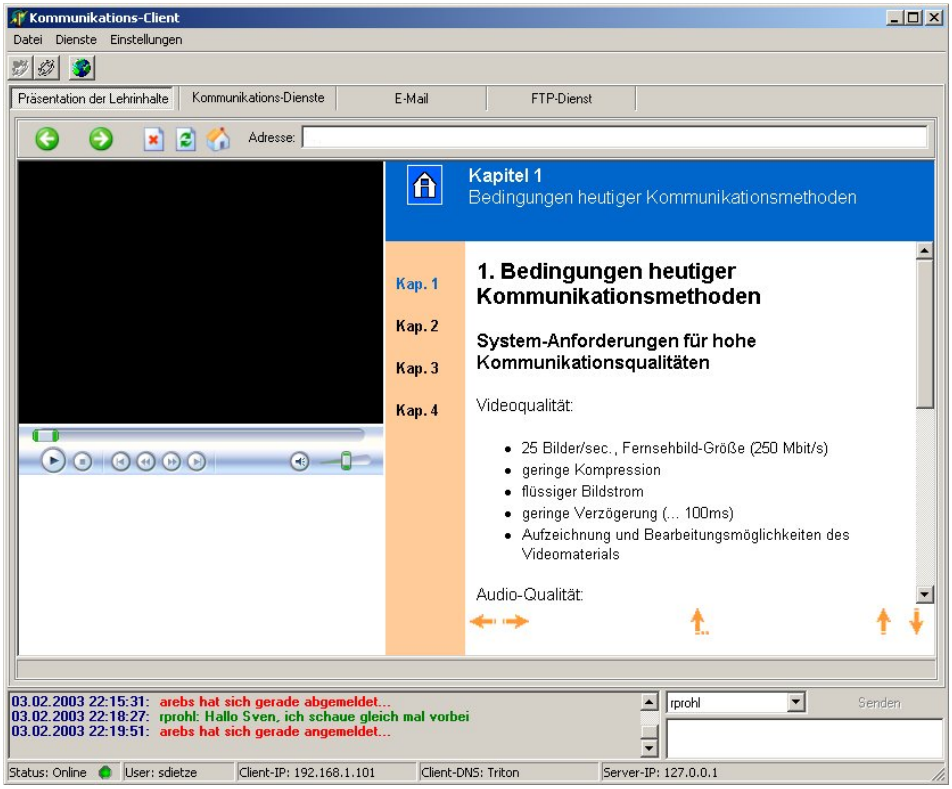

Abbildung 4.4: Präsentationsdienst im Client-Modul

Die aus der *Abbildung 4.4* ersichtliche Menüleiste und Werkzeugleiste des Client-Moduls bieten dabei die nachfolgend beschriebenen Funktionalitäten:

- Menüpunkt **Datei**
  - Die Menüoption **Verbindung herstellen** bringt dem Anwender einen Eingabedialog zur Ansicht, in welchen das zur Anmeldung an den zentralen Server benötigte Login und Passwort eingegeben werden müssen. Durch Bestätigung dieses Dialogfeldes über den darin enthaltenen OK-Button wird die Client-Anwendung in den ONLINE-Modus versetzt und die Verbindung zum Server aufgebaut. Analog zur

hier beschriebenen Menüoption kann auch der erste Tool-Button der Werkzeugleiste verwendet werden.

- Mit Hilfe der Menüoption **Verbindung trennen** wird eine bestehende Verbindung zum zentralen Server beendet und das Client-Modul wechselt in den OFFLINE-Modus. Dabei werden vor diesem Verbindungsabbau auch alle eventuell noch bestehenden Verbindungen zu den Dienst-Server-Modulen des Systems geschlossen. Analog zur hier beschriebenen Menüoption kann auch der zweite Tool-Button der Werkzeugleiste verwendet werden.
- Durch Auswahl der Menüoption **Beenden** wird das Client-Modul geschlossen. Dabei werden eventuell bestehende Verbindungen zum zentralen Server beziehungsweise den Dienst-Server-Modulen beendet.

- Menüpunkt **Dienste**
  - Über die Menüoptionen **Präsentation**, **Chat**, **Whiteboard** und **Videokonferenz** werden die grafischen Steuerelemente der jeweiligen Dienste zur Ansicht gebracht, welche in die Benutzeroberfläche des zentralen Clients integriert sind. Analog zu den hier beschriebenen Menüoptionen können auch die entsprechenden Button der aus *Abbildung 4.4* ersichtlichen Registerkarten verwendet werden, wobei die Dienste Chat, Whiteboard und Videokonferenz hierbei über die Registerkarte *Kommunikationsdienste* auswählbar sind. Der in der *Abbildung 4.5* dargestellte Screenshot macht dies deutlich.
  - Die Menüoptionen **eMail** und **FTP** dienen zum Aufruf der Dienste für eMail und FTP, wobei es sich bei beiden Diensten um eigenständige Programme handelt, deren Steuerelemente nicht in die Benutzeroberfläche des zentralen Clients integriert sind. Analog zu den hier beschriebenen Menüoptionen können auch die Button *E-Mail* und *FTP* der aus *Abbildung 4.4* ersichtlichen Registerkarten verwendet werden.

- Menüpunkt **Einstellungen**
  - Bei Auswahl der Menüoption **Server-Adresse** wird dem Anwender eine Eingabebox zur Ansicht gebracht, über welche die IP-Adresse des zentralen Servers festgelegt wird. Während einer bestehenden Verbindung zum zentralen Server hat die Änderung dieser IP-Adresse keine Auswirkung, sondern erst beim nächsten Verbindungsaufbau zu diesem. Analog zur hier beschriebenen Menüoption kann auch der dritte Tool-Button der Werkzeugleiste verwendet werden.

Die Benutzeroberfläche des Client-Moduls gliedert sich ebenfalls in drei Bereiche. Während im linken unteren Bereich vom Server eingehende Statusmeldungen sowie für den aktuell an-

gemeldeten Nutzer bestimmte Kurznachrichten angezeigt werden, dient der rechte untere Bereich zum Verfassen solcher Kurznachrichten. Hierzu steht neben dem Eingabefeld auch eine Auswahlbox mit den Kennungen der angemeldeten Nutzer bereit, aus welcher der Empfänger der Nachrricht ausgewählt wird. Der Hauptbereich bietet auch in diesem Modul über dafür verfügbare Registerkarten Zugriff auf die einzelnen Dienste, wobei in *Abbildung 4.4* der Dienst zur Präsentation der Vorlesungsinhalte dargestellt wird. Die Vorlesungsinhalte werden hierbei über einen integrierten Web-Browser angezeigt, der im gezeigten Screenshot gerade Vorlesungsinhalte mit integrierten Audio/Video-Inhalten darstellt. Die Aufteilung des durch diesen Web-Browser dargestellten HTML-Dokumentes ist nach der *Abbildung 3.6* gestaltet, welche das hierzu gehörige Konzept beinhaltet.

Bei Auswahl der Registerkarte *Kommunikationsdienste* eröffnet die Benutzeroberfläche dem Anwender eine weitere Auswahl an Registerkarten, welche die Steuerelemente der Zusatzdienste für Chat, Whiteboard und Videokonferenz zur Ansicht bringen. Der in *Abbildung 4.5* dargestellt Screenshot zeigt hierbei im Hauptbereich des Client-Moduls stellvertretend für diese Zusatzdienste das Nutzerinterface zur Verwendung des Whiteboard-Dienstes.

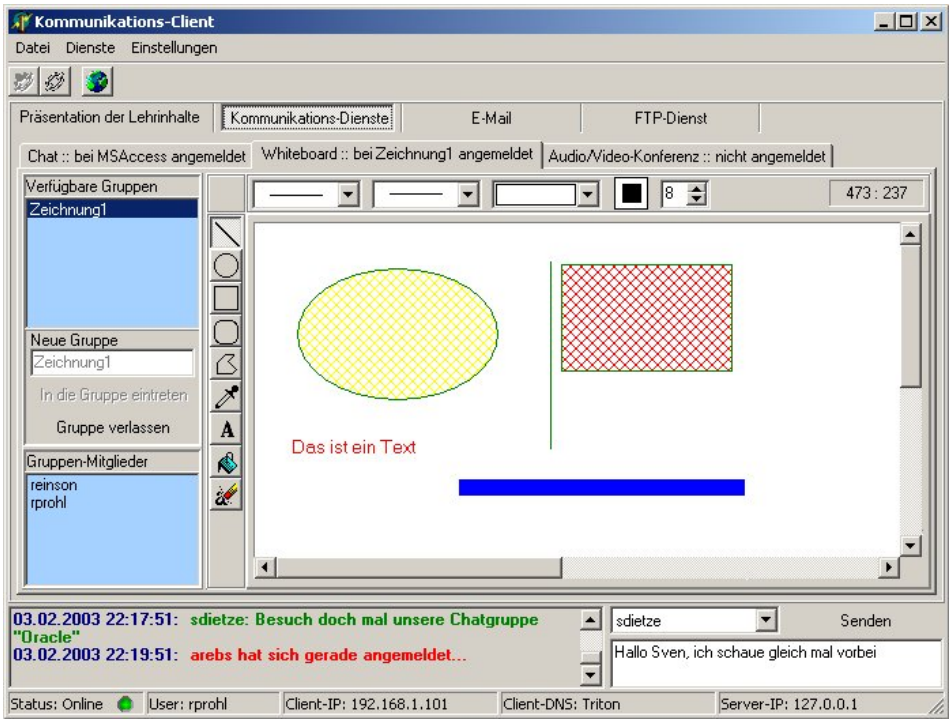

Abbildung 4.5: Whiteboard-Dienst im Client-Modul

Der Hauptbereich gliedert sich in zwei Bereiche, wobei im linken Bereich die am Whiteboard-Dienst angemeldeten Gruppen sowie deren zugehörige Nutzer in Listenform angezeigt werden. Der rechte Bereich hingegen beinhaltet die Arbeitsfläche für die Zeichenoperationen

sowie zwei hierzu gehörige Werkzeugleisten, welche links und oberhalb der Zeichenfläche positioniert sind. Während die links positionierte Werkzeugleiste zur Auswahl des Zeichenwerkzeugs dient, werden mit Hilfe der oberen Werkzeugleiste Einstellungen bezüglich Linienart, Linienstärke, Füllmuster, Stiftfarbe, Füllfarbe, Schriftfarbe und Schriftgröße vorgenommen.

Zur Eröffnung einer Gruppe steht ein Eingabefeld zur Verfügung, welches den gewünschten Gruppennamen entgegennimmt und dessen Eingabe durch das Drücken einer hierfür vorhandenen Schaltfläche *In die Gruppe eintreten* bestätigt wird. Der Eintritt in eine bereits bestehende Gruppe hingegen erfolgt entweder durch einen Doppelklick auf den zugehörigen Eintrag der Gruppenliste oder die Markierung dieser in selbiger Liste mit darauffolgender Bestätigung der Beitrittsabsicht durch Drücken der Schaltfäche *In die Gruppe eintreten*. Für das Verlassen einer Gruppe ist die Schaltfläche *Gruppe verlassen* vorhanden.

Kapitel 5

Ergebnisse

Um Aussagen bezüglich der realen Einsatzfähigkeit des hier entwickelten Systems treffen zu können, werden entsprechende Leistungstests auf unterschiedlichen Client-Maschinen durchgeführt. Diese Tests geben vor allem auch Aufschluss darüber, ob beziehungsweise inwieweit das Gesamtkonzept der vorliegenden Arbeit hinsichtlich der Verfügbarkeit von Hauptdienst und Zusatzdiensten für Client-Rechner mit schmalbandiger Netzanbindung tatsächlich umgesetzt wird. Zudem wird ein Vergleich zwischen dem im Rahmen dieser Arbeit entstandenen Distance-Learning-System und dem System OVID gezogen, dem die anhand des durchgeführten Leistungstests gewonnenen Erkenntnisse als Grundlage dienen, und welcher die jeweils abweichenden Konzepte beider Systeme kurz gegenüberstellt sowie im Ergebnis bewertet. Zunächst folgt jedoch erst einmal die Vorstellung der eingangs beschriebenen Leistungstests und deren Resultate.

5.1 Leistungstest

Die Client-Anwendung des hier entwickelten Distance-Learning-Systems wird im Rahmen dieses Leistungstests auf drei unterschiedlichen Rechnersystemen getestet, welche sich bezüglich Rechenleistung, verfügbarem Arbeitsspeicher, verwendeter Browser-Software sowie Betriebssystem unterscheiden. Der zentrale Server läuft auf einem vom Streaming-Server und Web-Server separat arbeitenden Rechnersystem, wobei als Web-Server ein APACHE HTTP-Server verwendet wird. Die Einzelheiten über die hierbei eingesetzte Hard- und Software werden nachfolgend aufgelistet:

- Client 1
  - *CPU*: Intel Pentium^{TM}III 1200 MHz
  - *RAM*: 512 MB
  - *Festplatte*: 40 GB

  - *Betriebssystem*: Windows XP™Professional
  - *Browser*: Internet Explorer™6.0

- Client 2
  - *CPU*: AMD Athlon™800 MHz
  - *RAM*: 512 MB
  - *Festplatte*: 30 GB
  - *Betriebssystem*: Windows XP™Professional
  - *Browser*: Internet Explorer™6.0

- Client 3
  - *CPU*: Intel Pentium™166 MHz
  - *RAM*: 32 MB
  - *Festplatte*: 2.1 GB
  - *Betriebssystem*: Windows 98™
  - *Browser*: Internet Explorer™5.0

- Server-Rechner (zentraler Server)
  - *CPU*: AMD Athlon™1400 MHz
  - *RAM*: 512 MB
  - *Festplatte*: 30 GB
  - *Betriebssystem*: Windows XP™Professional.

Als Netzanbindung kommen für die einzelnen Client-Rechner während des Testlaufes sowohl eine 10 Mbit LAN-Verbindung als auch eine 56 Kbit Modem-Verbindung zum Einsatz, um vor allem die Nutzbarkeit der Dienste zur Präsentation der Vorlesungsinhalte beziehungsweise zur Videokonferenz hinsichtlich der verfügbaren Bandbreite beurteilen zu können. Die Verbindung der LAN-Rechner untereinander erfolgt über einen SOHO-Router mit integrierter Switch, wobei dieser SOHO-Router seinerseits eine Anbindung zum Internet mit einer Bandbreite von 256 Kbit/s bereitstellt. Die hier beschriebene Netzwerkkonfiguration wird in *Abbildung 5.1* dargestellt, wobei der in dieser Darstellung abgebildete Rechner *Client 1* bei einigen Testläufen alternativ zur dargestellten Modem-Anbindung über eine LAN-Anbindung mit dem zentralen Server verbunden ist.

Für die Client-Anwendung wird zum Zweck der hier durchgeführten Tests ein Simulationsmodul verwendet, welches im Rahmen dieser Arbeit entwickelt wurde. Dieses generiert

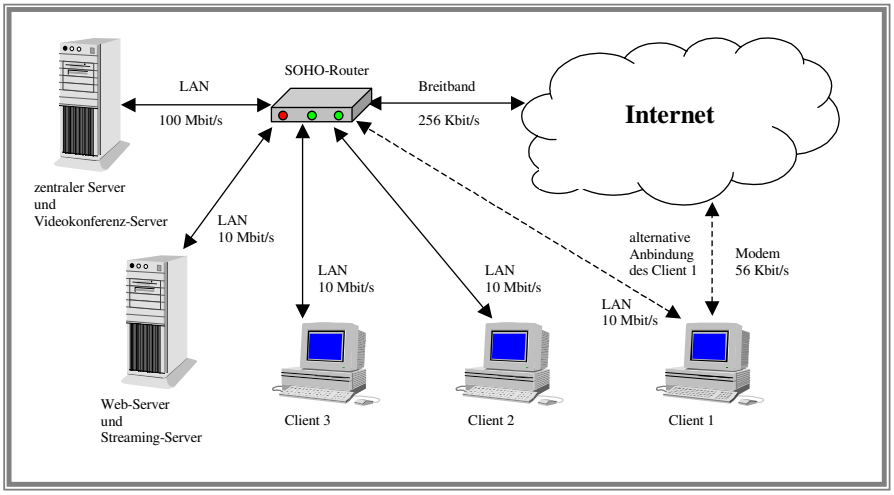

Abbildung 5.1: Netzwerkseitige Konfiguration während des Testlaufes

selbstständig für die Dienste Chat und Whiteboard kontinuierlich Chat-Nachrichten beziehungsweise Zeichenoperationen und stellt sie diesen Diensten zur Verfügung. Nach der Anmeldung an eine Gruppe beginnen diese Dienste mit dem Versenden dieser Daten an das jeweilige Dienst-Server-Modul, um auf diese Weise eine reale Nutzung dieser Dienste für mehrere Nutzer im Rahmen einer Gruppenarbeit zu simulieren. Bei der Simulation eines Chat-Teilnehmers wird hierbei eine Nachricht zufälliger Länge generiert und vom Chat-Client versendet, welche sich aus 3 bis 50 Zeichen zusammensetzt. Der zeitliche Abstand zwischen dem Versenden solcher Chat-Nachrichten ist dabei so gewählt, dass für jeweils 3 Zeichen eine Sekunde berechnet wird. Analog hierzu werden für den Whiteboard-Client in einem zufälligen Abstand zwischen 2 und 5 Sekunden Zeichenoperationen generiert und von diesem versendet, welche ein Rechteck begrenzter Größe und Position beschreiben.

Für den Server-Rechner, auf welchem der zentrale Server ausgeführt wird, werden zum Zweck der Ermittlung der an diesem Rechner aufkommenden CPU-Auslastung verschiedene Szenarien simuliert, welche sich in der Anzahl der an einen Dienst angemeldeten Nutzer und der Art des verwendeten Dienstes unterscheiden. Die *Tabelle 5.1* zeigt die beim Testlauf ermittelten Werte. Hieraus wird ersichtlich, dass das Server-Modul das Rechnersystem bezüglich der CPU-Last entsprechend der simulierten Szenarien nur geringfügig belastet. Nach erfolgreicher Anmeldung eines Nutzers an das System wird dem Vorlesungsdienst bereits die Adresse der Startseite übermittelt, so dass sich alle weiteren Operationen zum Datenaustausch zwischen Server-Modul und zentralem Dienst auf die Aktualisierung von Gruppen- und Nutzerlisten beschränkt. Die in *Tabelle 5.1* aufgeführten Werte für die Zusatzdienste Chat und Whiteboard beinhalten, dass sich alle Anwender, die den entsprechenden Dienst nutzen, an derselben Gruppe angemeldet haben. Hieraus resultiert, dass das Dienst-Server-Modul alle bei ihm eingehenden Daten auch an alle laut Szenario angemeldeten Teilnehmer weiterleiten muss, und

CPU-Auslastung des Servers			
genutzer Dienst	**1 bis 4 Nutzer**	**7 Nutzer**	**10 Nutzer**
zentraler Dienst	$< 1\%$	$< 1\%$	$< 1\%$
Chat	max. 1%	max. 2%	max. 2%
Whiteboard	max. 1%	max. 2%	max. 2%
Chat + Whiteboard	max. 2%	max. 4%	max. 4%

Tabelle 5.1: CPU-Auslastung des Servers in Abhängigkeit von der Nutzeranzahl und den von ihnen verwendeten Diensten

somit eine maximale Server-seitige Belastung entsprechend der Testsimulation entsteht. Die in *Tabelle 5.1* aufgeführte gleichzeitge Nutzung der Dienste für Chat und Whiteboard dient lediglich zur Ermittlung der CPU-Auslastung bei erhöhter Beanspruchung des Server-Rechners und gleicher Anzahl von Nutzern, denn in der Praxis wird es einem Anwender nicht möglich sein, zum gleichen Zeitpunkt eine Chat-Nachricht zu verfassen und eine Zeichenoperation durchzuführen.

Analog zur Prozessorlast am Server-Rechner wird in *Tabelle 5.2* die während der Testreihe auftretende CPU-Auslastung auf Seiten der Client-Rechner anhand der verfügbaren Dienste dargestellt, wobei für alle ermittelten Messwerte gilt, dass immer jeweils nur einer der in dieser Tabelle aufgeführten Dienste zum Zeitpunkt der Messung aktiv ist. Hieraus wird ersichtlich, dass die Dienste, welche nicht die Übermittlung von Audio/Video-Daten beinhalten, auf jedem der drei Testrechner problemlos genutzt werden können. Hierbei treten beim Datenaustausch mit dem Server-Modul beziehungsweise den darin integrierten Dienst-Server-Modulen keine bemerkbaren Verzögerungen auf. Anders verhält es sich dagegen bei Nutzung des Dienstes zur Präsentation der Vorlesungsinhalte, wenn die dabei geladenen HTML-Dokumente Audio/Video-Daten enthalten. Im Testlauf werden hierzu drei verschiedene Video-Streams verwendet, welche zwar denselben Inhalt präsentieren, sich aber in ihrer Qualität, und somit auch in ihrem Datenaufkommen, stark unterscheiden. Diese Streams beinhalten hierbei ein Video mit einer Spieldauer von 12 Sekunden, welches auch Audiodaten enthält. Während die beiden qualitativ höherwertigen Streams mit einer Gesamt-Bitrate von 548 Kbit/s beziehungsweise 1128 Kbit/s den Videoinhalt in einer Auflösung von 320 x 240 Pixeln und einer Bildwiederholrate von 30 Bildern je Sekunde beinhalten, wird der qualitativ niederwertigste Stream dieser Testreihe mit einer Gesamt-Bitrate von 38 Kbit/s in einer Auflösung von 160 x 120 Pixeln und einer Bildwiederholrate von 15 Bildern je Sekunde dargestellt. Die beiden höherwertigen Streams besitzen zwar die gleichen Eigenschaften bezüglich der Bildauflösung und Bildwiederholrate, jedoch unterliegt der Stream mit einer Gesamt-Bitrate von 1128 Kbit/s einer geringeren Komprimierung und bietet somit eine höhere Wiedergabequalität. Für alle drei Streams wird als Video-Codec *Windows Media Video 9* beziehungsweise als Audio-Codec *Win-*

CPU-Auslastung der Clients			
genutzer Dienst	**Client 1** (1.2 GHz)	**Client 2** (800 MHz)	**Client 3** (166 MHz)
Vorlesungsdienst ohne Audio/Video-Daten	max. 5%	max. 6%	max. 14%
Vorlesungsdienst mit Audio/Video-Daten 38 Kbps / 15 fps	max. 35% fließender Bildlauf	max. 40% fließender Bildlauf	80-100% fließender Bildlauf
Vorlesungsdienst mit Audio/Video-Daten 548 Kbps / 30 fps	max. 35% fließender Bildlauf	max. 40% fließender Bildlauf	100% kein fließender Bildlauf
Vorlesungsdienst mit Audio/Video-Daten 1128 Kbps / 30 fps	max. 45% fließender Bildlauf	max. 48% fließender Bildlauf	-
Chat Whiteboard	max. 1% max. 1%	max. 1% max. 1%	max. 2% max. 2%
Videokonferenz QCIF (176 x 144 Pixel)	max. 17% fließender Bildlauf	max. 20% fließender Bildlauf	100% kein fließender Bildlauf

Tabelle 5.2: CPU-Auslastung der Clients bei Nutzung verschiedener Dienste

dows Media Audio 9 verwendet. Während die ersten beiden Testrechner *Client 1* und *Client 2* beim Lesen und Darstellen dieser drei Streams einen flüssigen Bildlauf ermöglichen, weist der Testrechner *Client 3* bei der Darstellung des Streams mit einer Gesamt-Bitrate von 38 Kbit/s bereits eine hohe Auslastung der CPU auf, wobei der Bildlauf aber immer noch flüssig ist. Bei der Darstellung des qualitativ höherwertigen Streams mit einer Gesamt-Bitrate von 548 Kbit/s hingegen ist die CPU des Rechners *Client 3* überlastet, so dass auch kein fließender Bildlauf mehr gegeben ist. Ein Testlauf zur Ermittlung der Prozessorlast des Rechners *Client 3* für den qualitativ höchstwertigen Stream mit einer Gesamt-Bitrate von 1128 Kbit/s erübrigt sich, da dieses Rechnersystem bereits beim Darstellen des Stream mit einer Gesamt-Bitrate von 548 Kbit/s überlastet ist.

Bei der Nutzung des Videokonferenz-Dienstes zeigt sich ein vergleichbares Verhalten, wie es bei der Nutzung des Dienstes zur Präsentation der Vorlesungsinhalte mit integrierten Audio/Video-Inhalten zu beobachten ist. Während die Testrechner *Client 1* und *Client 2* bei Nutzung des Videokonferenz-Dienstes einen fließenden Bildlauf bei einer CPU-Last von maximal 17% beziehungsweise 20% ermöglichen, ist die Verwendung dieses Dienstes entsprechend den Erwartungen beim Testrechner *Client 3* infolge einer Überlastung der CPU und den daraus

resultierenden Unterbrechungen bei der Wiedergabe der Audio/Video-Daten nicht möglich.

Zur Bestimmung der Ursachen für die jeweils ermittelte CPU-Auslastung bezüglich der Nutzung von Diensten mit Audio/Video-Inhalten auf den drei Client-Rechner werden die beim Testlauf erhaltenen Werte im Folgenden näher betrachtet und ausgewertet. Dabei wird außerdem auch die auftretende Netzlast für die jeweiligen Dienste sowie unterschiedlichen Konfigurationen untersucht und bewertet.

In *Abbildung 5.2* werden die Prozessorlast sowie die Ereignisse bezüglich Download und Wiedergabe des 38 Kbit/s umfassenden Audio/Video-Streams für die drei Client-Rechner im zeitlichen Verlauf am Beispiel des Vorlesungsdienstes detailiert dargestellt, wobei die Netzanbindung der Clients eine Bandbreite von 10 Mbit/s bereitstellt.

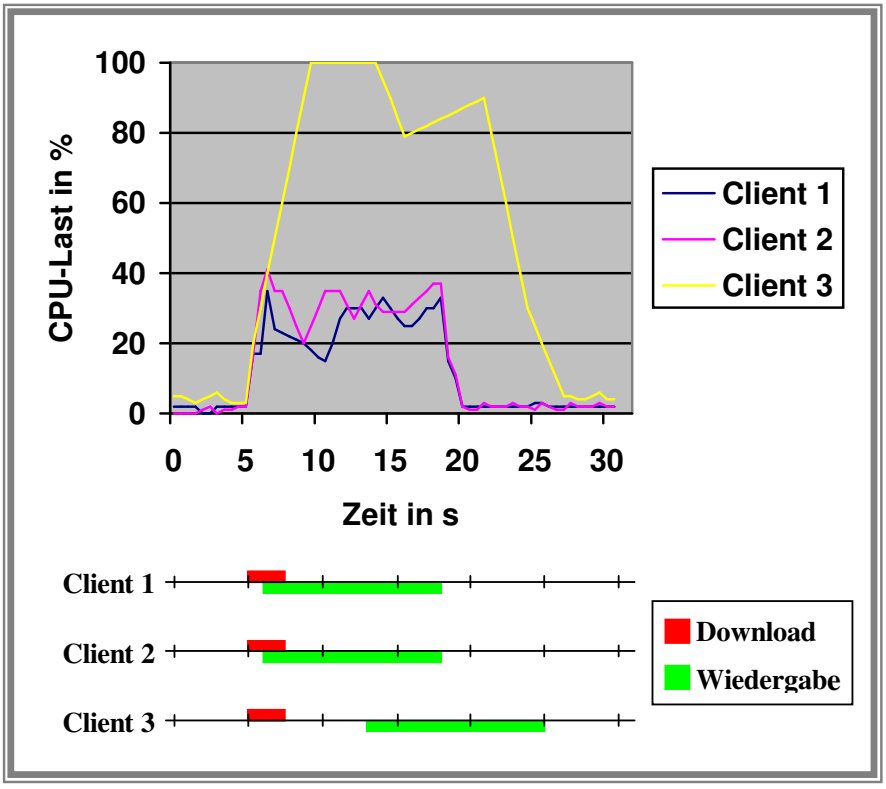

Abbildung 5.2: CPU-Auslastung und Ereignisverlauf für den Vorlesungsdienst mit Audio/Video-Daten (Gesamt-Bitrate 38 Kbit/s) bei Verwendung einer 10 Mbit LAN-Verbindung

Der Aufruf des HTML-Dokumentes mit dem integrierten Audio/Video-Inhalt erfolgt im dargestellten Diagramm in Bezug auf die Zeitachse für alle drei Client-Rechner bei 5 Sekunden. Während der zeitliche Abstand zwischen dem Aufruf des HTML-Dokumentes und dem Beginn der Wiedergabe der Audio/Video-Daten bei den Testrechnern *Client 1* und *Client 2* ca. 1 Sekunde beträgt, benötigt *Client 3* hierfür etwa 8 Sekunden. In diesem Zeitraum werden das ActiveX-Control des im HTML-Dokument eingebetteten Media Players initialisiert und emp-

fangene Audio/Video-Daten des Streams gepuffert sowie zur Wiedergabe aufbereitet. Aus dem Diagramm der *Abbildung 5.2* ist ersichtlich, dass in diesem Zeitraum die CPU-Last bereits den in der *Tabelle 5.2* für den 38 Kbit Stream jeweils angegebenen Maximalwert erreicht. Während sich die CPU-Last der Testrechner *Client 1* und *Client 2* bereits 1 Sekunde nach dem Aufruf des HTML-Dokumentes wieder verringert und danach unterhalb des in *Tabelle 5.2* angegebenen Maximalwertes bewegt, erfolgt diese Entlastung für *Client 3* erst nach ca. 9 Sekunden. Um die Ursache hierfür bestimmen zu können, werden zwei weitere Tests durchgeführt, welche die jeweilige CPU-Last unter den gegebenen Bedingungen getrennt für den Download des Audio/Video-Streams und dessen Wiedergabe ermitteln. Hierzu wird die Stream-Datei per Browser auf den Client-Rechner heruntergeladen beziehungsweise das HTML-Dokument und der darin eingebundene Audio/Video-Stream auf dem jeweilgen Client-Rechner lokal aufgerufen. Diese beiden Tests werden für alle drei Client-Rechner durchgeführt und die Ergebnisse in den *Abbildungen 5.3 und 5.4* dargestellt.

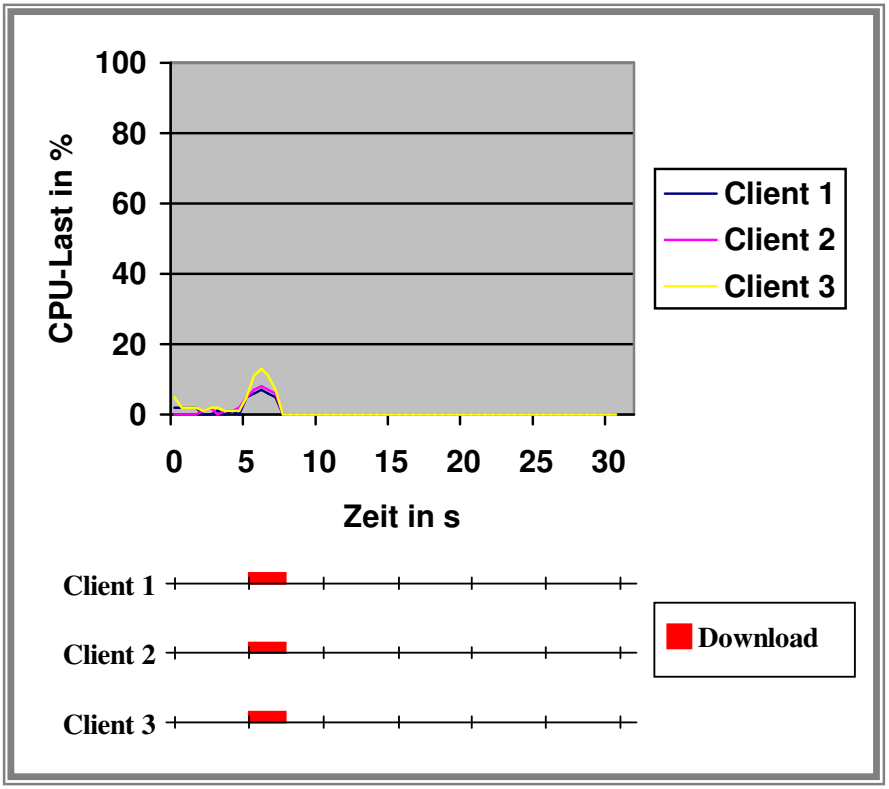

Abbildung 5.3: CPU-Auslastung und Ereignisverlauf für den Download der Audio/Video-Daten (Gesamt-Bitrate 38 Kbit/s) ohne Wiedergabe durch den Media Player

Wie die Messwerte aus *Abbildung 5.3* erkennen lassen, wird die CPU der einzelnen Client-Rechner beim Download der Stream-Datei mit Werten von 7% bis 13% nur teilweise belastet. Die Wiedergabe eines lokalen Streams hingegen verursacht eine CPU-Last, welche jener bei

der Wiedergabe eines per LAN-Anbindung übertragenen Streams entspricht. Die beim Test ermittelten und in *Abbildung 5.4* dargestellten Messwerte machen dies deutlich. Auch hier wird die CPU des Testrechners *Client 3* bis zu 100% belastet und die Wiedergabe gegenüber den anderen beiden Testrechnern *Client 1* und *Client 2* deutlich verzögert.

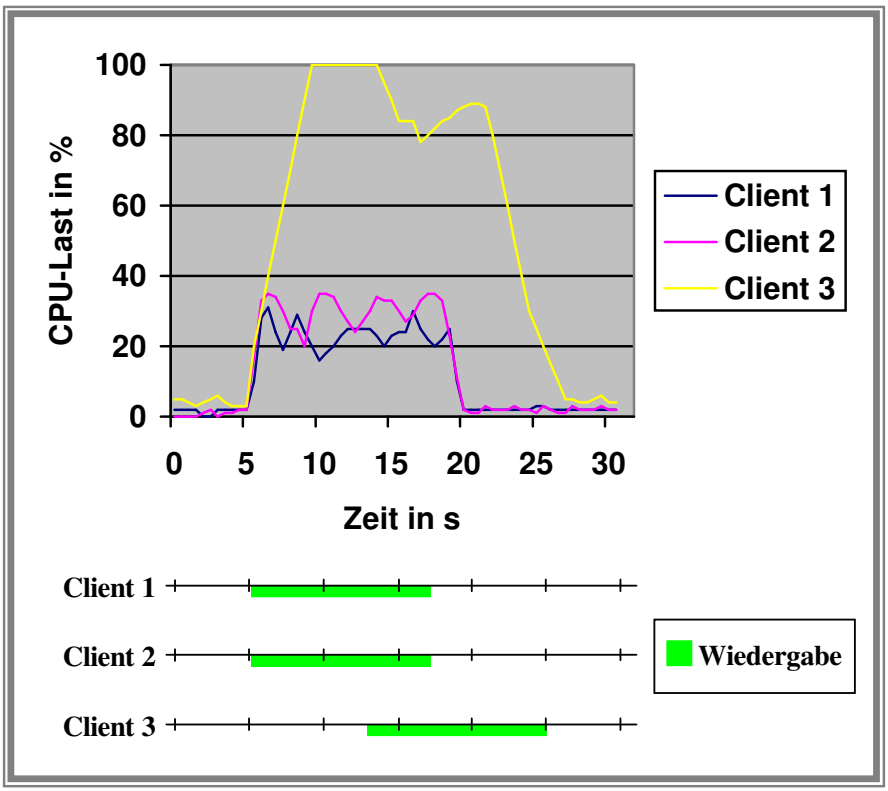

Abbildung 5.4: CPU-Auslastung und Ereignisverlauf für den Vorlesungsdienst mit Audio/Video-Daten (Gesamt-Bitrate 38 Kbit/s) bei Verwendung eines lokalen Streams (kein Download)

Bezugnehmend auf *Abbildung 5.2* kann also aufgrund der ermittelten und in den *Abbildungen 5.3 und 5.4* dargestellten Werte gesagt werden, dass *Client 3* aufgrund des zu klein dimensionierten Arbeitsspeichers sowie der im Vergleich zu den anderen beiden Testrechnern geringeren Rechenleistung und einer dabei auftretenden CPU-Last von 100% mehr Zeit benötigt, um die beim Streaming vor der Wiedergabe anfallenden Operationen auszuführen. Nach diesem Initialisierungsvorgang wird die CPU zwar immer noch bis zu 90% belastet, jedoch ist die verfügbare Rechenzeit damit immer noch ausreichend und gewährleistet somit einen fließenden Bildlauf.

Die durchschnittliche Netzwerkauslastung auf Client-Seite bei Verwendung einer 10 Mbit LAN-Anbindung wird für die 3 qualitativ verschiedenen Streams in *Abbildung 5.5* dargestellt, wobei der qualitativ niederwertigste Stream mit einer Gesamt-Bitrate von 38 Kbit/s bezüglich

der Diagramm-Zeitachse im Bereich von 5 bis 7.5 Sekunden abgebildet ist. Die Messwerte für den Stream mit einer Gesamt-Bitrate von 548 Kbit/s sind im Zeitbereich von 15 bis 19 Sekunden dargestellt, jene für den qualitativ höchstwertigen Stream im Bereich von 25 bis 31 Sekunden.

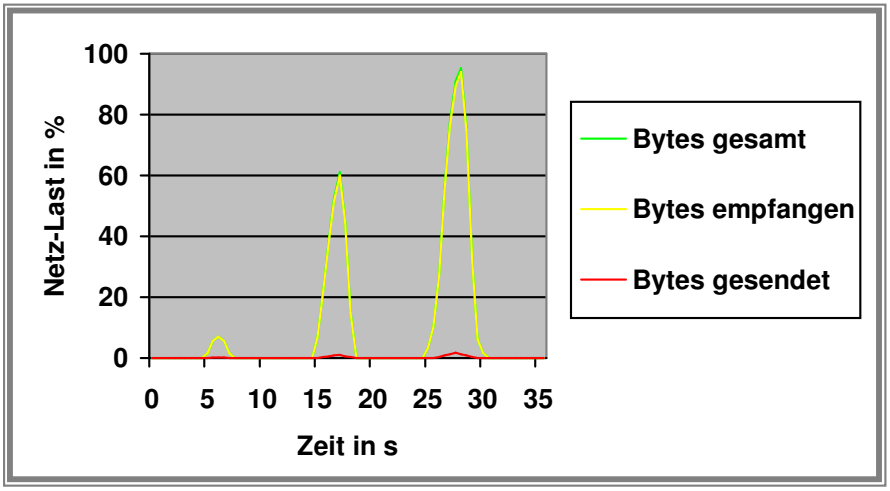

Abbildung 5.5: Durchschnittliche Netzwerkauslastung für den Vorlesungsdienst mit Audio/Video-Daten (Gesamt-Bitrate 38 Kbit/s, 548 Kbit/s und 1128 Kbit/s) bei Verwendung einer 10 Mbit LAN-Verbindung für *Client 1* (v.l.n.r)

Wie aus den Messwerten der *Abbildung 5.5* hervorgeht, beträgt die bei den Tests ermittelte Zeitdauer zur Übertragung der drei einzelnen Streams über eine 10 Mbit LAN-Verbindung 2.5, 4 beziehungsweise 6 Sekunden. Vergleicht man die für den 38 Kbit/s umfassenden Stream ermittelte Übertragungsdauer von 2.5 Sekunden mit dem zeitlichen Abstand zwischen dem Aufruf des HTML-Dokumentes bis hin zur Wiedergabe dieses Streams, so wird ersichtlich, dass die Wiedergabe Streaming-typisch bereits nach der Pufferung eines Teils des gesamten Streams erfolgt und nicht auf den vollständigen Download der Daten warten muss. Ausnahme hierbei bildet der Testrechner *Client 3*. Dieser kann diesen Vorteil nicht nutzen, da er infolge der zu geringen Rechenleistung und des zu klein dimensionierten Arbeitsspeichers für die Ausführung der anfallenden Operationen entsprechend mehr Zeit benötigt und die Wiedergabe deshalb mit einer zeitlichen Verzögerung von ca. 8 Sekunden beginnt, also zu einem Zeitpunkt, an dem die Streaming-Daten bereits vollständig übertragen sind.

In *Abbildung 5.6* werden analog zur *Abbildung 5.2* die Auslastung der CPU sowie die Ereignisse bezüglich Download und Wiedergabe des Streams für die drei Testrechner beschrieben, nur dass hier anstatt des Streams mit einer Gesamt-Bitrate von 38 Kbit/s jener mit 548 Kbit/s dargestellt wird. Die Messwerte für *Client 1* und *Client 2* zeigen, dass sich die CPU-Last dieser beiden Testrechner bei der Wiedergabe des Streams mit einer Gesamt-Bitrate von 548 Kbit/s unwesentlich von jener des Streams mit 38 Kbit/s unterscheidet. Das dabei aufgrund der

höheren Bildauflösung sowie höherer Bildwiederholrate entsprechend größere Datenaufkommen und die damit verbundenen Operationen zur Software-seitigen Dekodierung dieser Daten verursachen also für die hier betrachteten Prozessoren keine erkennbare Mehrbelastung. Die CPU-Last erreicht hierbei zu Beginn der Datenübertragung ebenfalls den hierfür in *Tabelle 5.2* angegebenen Maximalwert, bevor sie sich kurzzeitig wieder verringert und mit dem um ca. 2 Sekunden verzögerten Wiedergabebeginn erneut ansteigt. Die gegenüber dem 38 Kbit/s umfassenden Stream angewachsene Verzögerung des Wiedergabebeginns begründet sich aus der längeren Übertragungsdauer der Streaming-Daten, welche anhand der beim Test ermittelten und in *Abbildung 5.5* dargestellten Werte ca. 4 Sekunden beträgt. Es muss also der Gesamt-Bitrate des Streams entsprechend auch eine größere Menge an Daten gepuffert werden, bevor die Wiedergabe erfolgt.

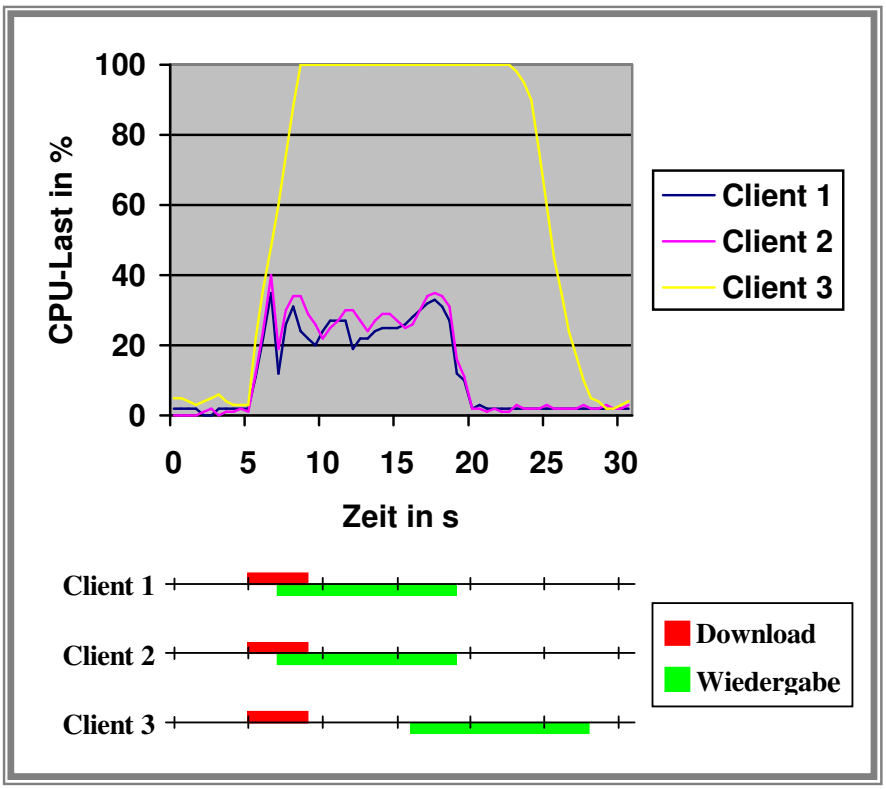

Abbildung 5.6: CPU-Auslastung und Ereignisverlauf für den Vorlesungsdienst mit Audio/Video-Daten (Gesamt-Bitrate 548 Kbit/s) bei Verwendung einer 10 Mbit LAN-Verbindung

Anders verhält sich dagegen der Testrechner *Client 3*. Im Gegensatz zu der in *Abbildung 5.2* dargestellten Kennlinie dieses Rechners, welche die Wiedergabe des Streams mit einer Gesamt-Bitrate von 38 Kbit/s beschreibt und eine Verringerung der Auslastung der CPU nach der Initialisierungsphase erkennen lässt, liegt die CPU-Last bei der Wiedergabe des 548 Kbit/s umfassenden Streams dauerhaft bei 100% und verringert sich erst gegen Ende der Wiedergabe. Die Wiedergabe der Audio/Video-Daten beginnt aufgrund der in Anbetracht der aus-

zuführenden Operationen zu geringen Prozessorleistung erst ca. 11 Sekunden nach dem Aufruf des HTML-Dokumentes, was auf der dargestellten Zeitachse dem Wert von 16 Sekunden entspricht. Zu diesem Zeitpunkt ist der Stream bereits vollständig an *Client 3* übertragen.

Die ermittelten Werte bezüglich der CPU-Auslastung der Testrechner *Client 1* und *Client 2* bei der Übertragung und Wiedergabe des 1128 Kbit/s umfassenden Streams sind Gegenstand der grafischen Darstellung in *Abbildung 5.7*.

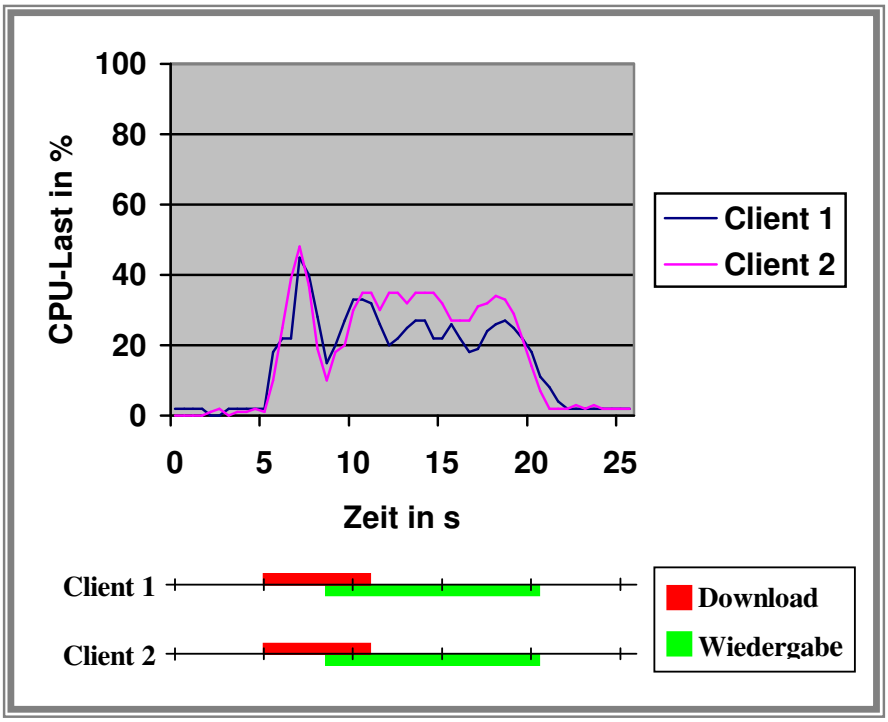

Abbildung 5.7: CPU-Auslastung und Ereignisverlauf für den Vorlesungsdienst mit Audio/Video-Daten (Gesamt-Bitrate 1128 Kbit/s) bei Verwendung einer 10 Mbit LAN-Verbindung

Mit dem Aufruf des HTML-Dokumentes steigt auch hier bei beiden Testrechnern die CPU-Last zunächst an und erreicht dabei den jeweils ermittelten Maximalwert, bevor sie sich in etwa gleicher Weise wieder verringert und mit dem Beginn der Wiedergabe des Streams erneut ansteigt. Dieser Zeitpunkt des Wiedergabebeginns liegt auf der dargestellten Zeitachse bei etwa 8.5 Sekunden, also 3.5 Sekunden nach dem Aufruf des HTML-Dokumentes. Diese gegenüber der beim 548 Kbit/s umfassenden Stream etwas größere Verzögerung des Wiedergabebeginns begründet sich durch das beim 1128 Kbit/s umfassenden Stream höhere Datenaufkommen. Die Übertragung des Streams vom Server zum Client-Rechner benötigt bei Verwendung einer 10 Mbit LAN-Verbindung ca. 6 Sekunden, wie aus den hierfür ermittelten Messwerten der *Abbildung 5.5* hervorgeht. Dementsprechend wird auch mehr Zeit benötigt, um den vor Wiedergabebeginn erforderlichen Umfang an Daten zu puffern.

Die CPU-Auslastung bei der Verwendung einer Modemverbindung mit einer maximalen Bandbreite von 56 Kbit/s unterscheidet sich von jener bei Verwendung einer 10 Mbit LAN-Verbindung lediglich im zeitlichen Verlauf, wie aus *Abbildung 5.8* hervorgeht. Den in dieser Abbildung dargestellten Messwerten liegt die Übertragung und Darstellung des für den Testlauf generierten Streams mit einer Gesamt-Bitrate von 38 Kbit/s und einer Spieldauer von 12 Sekunden zu Grunde. Als Testrechner wird in beiden Fällen *Client 1* verwendet. Während für beide Varianten der Netzanbindung der Aufruf des HTML-Dokumentes mit dem integrierten Audio/Video-Inhalt in Bezug auf die Zeitachse bei 5 Sekunden erfolgt, unterscheidet sich der Zeitpunkt der Wiedergabe deutlich. Bei Verwendung der LAN-Verbindung sind bereits nach 1 Sekunde Daten des Streams in ausreichender Menge gepuffert, so dass entsprechend der abgebildeten Zeitachse bei 6 Sekunden die Wiedergabe des Streams beginnt. Bei Verwendung der Modemverbindung hingegen beginnt die Wiedergabe erst 15 Sekunden nach dem Aufruf des HTML-Dokumentes mit integriertem Audio/Video-Inhalt. Nach anfänglichem Anstieg der CPU-Last beim Initialisierungsvorgang vermindert sich diese sofort wieder und steigt erst beim Wiedergabebeginn erneut deutlich an. Das Puffern der Daten belastet die CPU also nur gering.

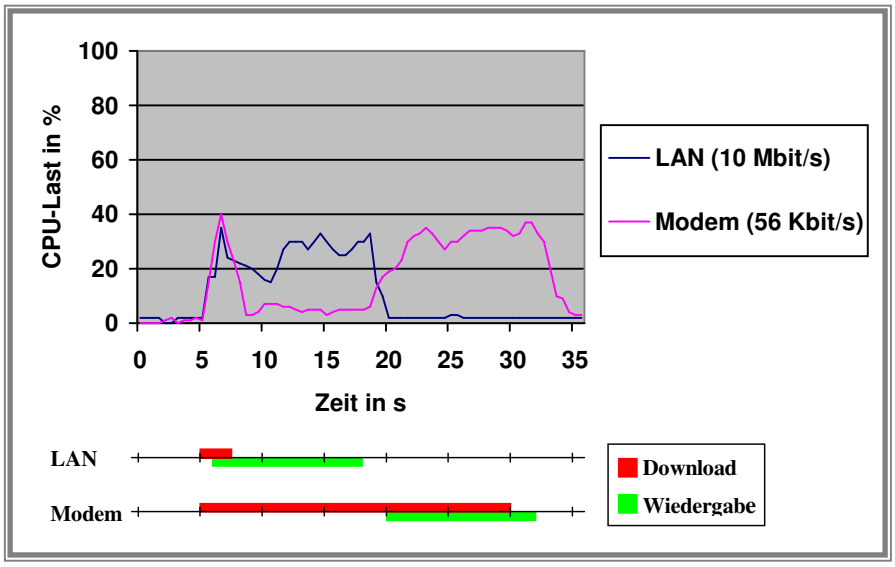

Abbildung 5.8: CPU-Auslastung und Ereignisverlauf für den Vorlesungsdienst mit Audio/Video-Daten (Gesamt-Bitrate 38 Kbit/s) bei Verwendung einer 56 Kbit Modemverbindung gegenüber einer 10 Mbit LAN-Verbindung für *Client 1*

Die in *Abbildung 5.9* dargestellte Client-seitige Netzwerkauslastung einer 56 Kbit Modemverbindung bei der Wiedergabe des 38 Kbit/s umfassenden Streams zeigt, dass hierbei nicht die gesamte Bandbreite genutzt wird. Der Stream wird dabei kontinuierlich übertragen, wobei die maximale Netzlast in etwa der Gesamt-Bitrate von 38 Kbit/s des Streams entspricht.

Die gegenüber der LAN-Verbindung wesentlich geringere Bandbreite begründet die zeitliche Verzögerung des Wiedergabebeginns von ca. 15 Sekunden, da hierbei zur Pufferung der dem Stream entsprechenden Datenmenge mehr Zeit benötigt wird. Wie die ermittelten Daten desweiteren zeigen, stellt die Bandbreite der Netzanbindung für die Nutzung des Präsentationsdienstes kein Hindernis dar. Der Einsatz eines Streaming-Servers gestattet durch die automatische Auswahl der qualitativ verschiedenen Streams in Abhängigkeit von der zur Verfügung stehenden Bandbreite auch die Verwendung einer schmalbandigen Modemverbindung, welche eine Datenübertragung im Umfang von maximal 56 Kbit/s gestattet.

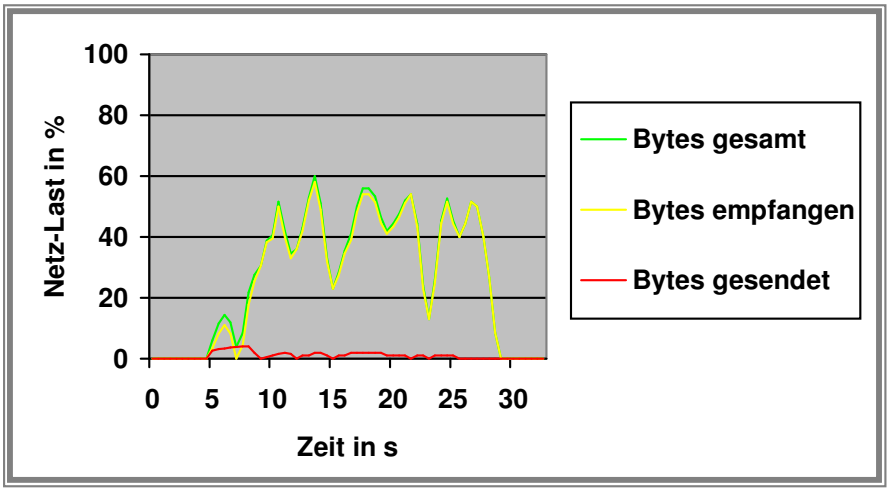

Abbildung 5.9: Netzwerkauslastung für den Vorlesungsdienst mit Audio/Video-Daten (Gesamt-Bitrate 38 Kbit/s) bei Verwendung einer 56 Kbit Modemverbindung für *Client 1*

Eine Harware-seitige Unterstützung bei der Dekodierung der im *Windows Media Format 9* übertragenen Streams, welche eine Entlastung der CPU des Rechners bezüglich der hierbei auszuführenden Operationen mit sich bringt, ist laut Aussagen von Microsoft®[MIC03-1] durch die nächste Generation von Grafikadaptern gegeben, welche die Technologie *DirectX®Video Acceleration (DxVA)* unterstützen. Hierbei verweist Microsoft®vor allem auf die Hersteller ATI®und NVidia®[MIC03-2].

Während die Nutzung des Videokonferenz-Dienstes für die Testrechner *Client 1* und *Client 2* eine CPU-Last von maximal 17% beziehungsweise 20% zur Folge hat, führt dieser Dienst beim Testrechner *Client 3* zu einer Überlastung der CPU, so dass ein fließender Bildlauf in diesem Fall nicht mehr gegeben ist. Die Ursache hierfür ist analog zu den für diesen Testrechner gewonnenen Erkenntnissen bezüglich des Präsentationsdienstes auf den hierfür mit 32 MB zu klein dimensionierten Arbeitsspeicher und die vergleichsweise geringe Rechenleistung des Pentium-Prozessors mit 166 MHz zurückzuführen. Die in *Abbildung 5.10* dargestellten Messwerte zeigen die ermittelten Daten für die drei Testrechner unter Verwendung einer 10 Mbit LAN-Verbindung. Die Überlastung des Prozessors von *Client 3* wird in dieser Darstel-

lung daraus ersichtlich, dass die CPU-Last über die gesamte Dauer der abgehaltenen Videokonferenz bei 100% liegt. Der Testrechner ist hierbei nicht in der Lage, die bei diesem Dienst auszuführenden Operationen zur Kodierung beziehungsweise Dekodierung der Daten in Echtzeit auszuführen.

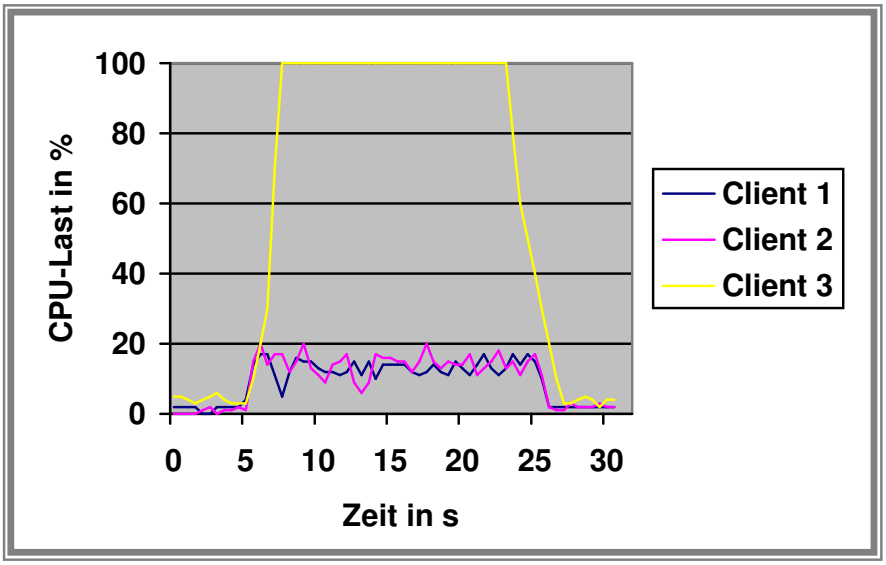

Abbildung 5.10: CPU-Auslastung für den Videokonferenz-Dienst (QCIF-Format) bei Verwendung einer 10 Mbit LAN-Verbindung

Die beim Videokonferenz-Dienst auf der Client-Seite auftretende Netzwerkbelastung unter Verwendung einer 10 Mbit LAN-Verbindung wird in *Abbildung 5.11* dargestellt. Hieraus ist zu erkennen, dass zur Übertragung der hierbei anfallenden Daten weniger als 2% der zur Verfügung stehenden Bandbreite verwendet werden. Der in dieser grafischen Darstellung ersichtliche Unterschied bezüglich der Datenmenge für gesendete und empfangene Bytes resultiert aus unterschiedlichen Bildquellen für beide Kommunikationspartner. Dabei kommt zur Geltung, dass größtenteils homogene Bildfolgen durch die angewendete Komprimierung in ihrem Datenumfang kleiner sind als Bildfolgen, welche Schwankungen bezüglich ihres dargestellten Inhaltes unterliegen.

Analog zur *Abbildung 5.11* wird auch in der *Abbildung 5.12* die beim Videokonferenz-Dienst auftretende Client-seitige Netzwerkauslastung beschrieben, nur dass hier anstatt der 10 Mbit LAN-Verbindung eine 56 Kbit Modemverbindung verwendet wird. Dabei wird die zur Verfügung stehende Bandbreite größtenteils ausgenutzt, wobei sich der Unterschied bezüglich der Datenmenge für gesendete und empfangene Bytes wiederum durch unterschiedliche Bildquellen für beide Kommunikationspartner erklärt.

Zur Ermittlung der Stabilität des Systems wird über einen Zeitraum von 3 Stunden die gleichzeitige Gruppenarbeit für die Zusatzdienste Chat und Whiteboard für insgesamt 5 Client-

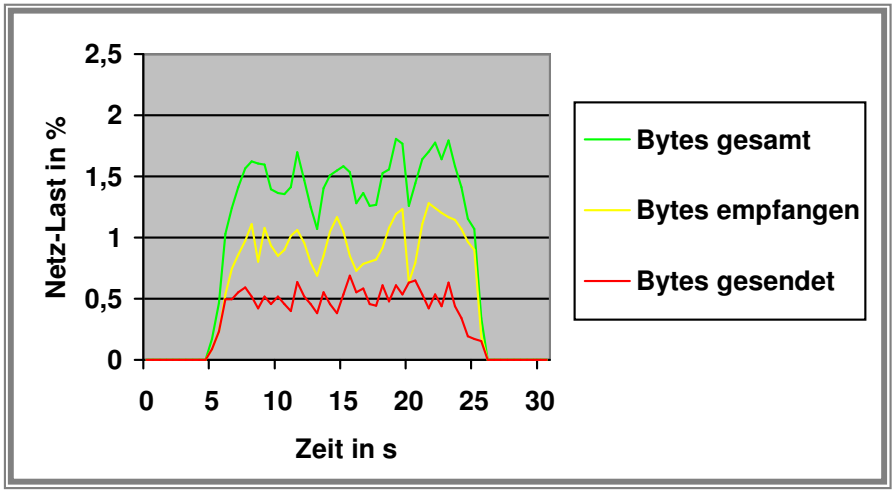

Abbildung 5.11: Netzwerkauslastung für den Videokonferenz-Dienst (QCIF) bei Verwendung einer 10 Mbit LAN-Verbindung für *Client 1*

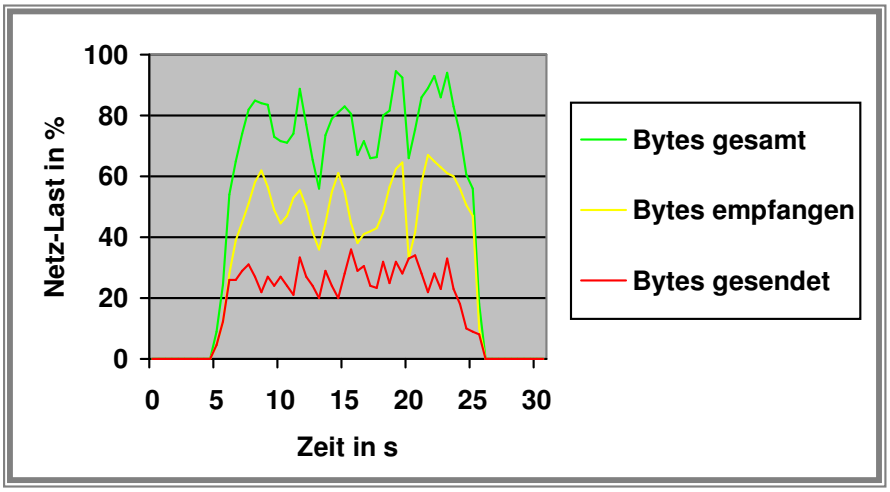

Abbildung 5.12: Netzwerkauslastung für den Videokonferenz-Dienst (QCIF) bei Verwendung einer 56 Kbit Modemverbindung für *Client 1*

Anwendungen simuliert, wobei wiederum alle Anwender derselben Gruppe des jeweiligen Dienstes angehören. Hierzu werden die 5 Anwender zu Beginn dieses Testlaufes an das System und die aufgezählten Zusatzdienste angemeldet und deren Gruppenarbeit nach 3 Stunden beendet. Die Simulation der Nutzereingaben erfolgt hierbei nach den zu Beginn dieses Kapitels festgelegten Vorgaben.

Während eines Testlaufes nach dem beschriebenen Muster sind keinerlei Fehler aufgetreten.

5.2 Vergleich mit OVID

Das Distance-Learning-Systems OVID und das hier entwickelte System GWOTS besitzen viele Gemeinsamkeiten, was die nachfolgend aufgeführten Eigenschaften beider Systeme auch verdeutlichen:

- Entwicklung des Systems für die Plattform von Microsoft Windows^{TM}
- entwickeltes System basiert auf der Client/Server-Architektur
- zentrale Verwaltung von Nutzern, Gruppen und Diensten
- Lehrinhalte werden in Form von HTML-Dokumenten abgespeichert und präsentiert
- Integration eines Browser-Moduls auf Client-Seite, dessen Kern mit dem im Anwendersystem installierten Web-Browser aktualisiert wird und das vielfältige Medien zur Präsentation der Lehrinhalte unterstützt
- integrierte Gruppenwerkzeuge in Form der Dienste für Chat, Whiteboard und Videokonferenz.

Der hauptsächliche Unterschied besteht neben einer vereinfachten Kommunikationsstruktur zum einheitlichen Sitzungsmanagement zwischen den Zusatzdienst-Modulen und dem zentralen Server sowie der Integration sämtlicher Benutzeroberflächen der einzelnen Dienste im zentralen Client-Modul vor allem in dem Grundsatz, dass alle zur Verfügung stehenden Dienste auch von Anwendern mit schmalbandiger Netzanbindung, etwa einer 56 Kbit/s Modemverbindung, genutzt werden können. Letzters betrifft neben dem Videokonferenz-Dienst insbesondere den Hauptdienst zur Präsentation der Lehrinhalte, da dieser den eigentlichen Anwendungszweck im Distance-Learning-System darstellt. Während dieser Dienst unter Verwendung von Audio/Video-Inhalten, welche in die präsentierte HTML-Seite integriert sind, im System OVID nur über eine Netzanbindung mit einer Bandbreite von mindestens 1.2 Mbit/s [ROS00] vom Anwender nutzbar ist, zeigen die Ergebnisse der Leistungstests beim hier entwickelten System, dass durch den Einsatz eines Streaming-Servers und der damit verbundenen Technologie auch Anwendern mit schmalbandiger Netzanbindung die Nutzung des Präsentationsdienstes ermöglicht wird. Die verminderte Qualität der übermittelteten Audio/Video-Daten, welche bei solchen schmalbandigen Netzanbindungen aufgrund kleiner Bildauflösungen und höherer Komprimierung entsteht, wird in Hinsicht auf die Nutzbarkeit dieses Dienstes in Kauf genommen.

Eine Vereinfachung des Zugriffs auf die bereitgestellten Lehrinhalte durch den Anwender wird im hier entwickelten System durch die Übermittlung einer Adresse zur Startseite realisiert, welche abhängig von dem zum Nutzer gehörigen Matrikel ist. Beim System OVID hingegen

wird dem Anwender an dieser Stelle eine Startseite präsentiert, welche die Verweise zu allen Lehrveranstaltungen enthält, die für sämtliche Fachbereiche und Matrikel auf dem System verfügbar sind.

Der Videokonferenz-Dienst bietet im hier betrachteten System durch die Verwendung von H.323 und das QCIF-Format zur Datenübermittlung nicht die hohe Übertragungsqualität, wie dies beim System OVID der Fall ist. Dafür ist er aber auch für Anwender mit schmalbandiger Netzanbindung nutzbar *(vgl. Kapitel 5.1)*, während beim OVID eine Bandbreite von mindestens 1.2 Mbit/s [ROS00] erforderlich ist. Wie die Leistungstests zeigen, werden dabei allerdings höhere Anforderungen an die Hardware des Client-Rechners in Bezug auf die Rechenleistung gestellt, als dies beim System OVID der Fall ist. Grund hierfür ist die Tatsache, dass die Kodierung beziehungsweise Dekodierung der bei diesem Dienst anfallenden Daten Software-seitig realisiert wird, während beim OVID eine Hardware-seitige Kodierung beziehungsweise Dekodierung per Videoschnittkarte erfolgt. Eine Hardware-seitige De/Kodierung ist zwar auch für H.323 möglich, allerdings wird hierzu spezielle Hardware benötigt, die bei den durchgeführten Tests nicht zur Verfügung stand. Eine solche Hardware-Komponente ist beispielsweise die USB-Kamera *ViaVideo* des Herstellers *Polycom*^{TM}, welche über einen integrierten Hardware-Codec für H.323 verfügt.

Das im vorliegenden System entwickelte Konzept zum Videokonferenz-Dienst bewirkt aufgrund der wesentlich geringeren Datenmengen, die bei der Übertragung der Audio/Video-Daten anfallen, auch eine Entlastung der Netzanbindung des Rechnersystems, auf welchem der Videokonferenz-Server (MCU) betrieben wird.

Durch die Berücksichtigung schmalbandiger Netzanbindungen in den Konzeptionen der im Rahmen dieser Arbeit beschriebenen Dienste wird vor allem der Praxisnähe bezüglich der vom einzelnen Anwender verwendeten Netzanbindung Rechnung getragen. Ein Distance-Learning-System muss seinen Nutzern, unabhängig von deren Standorten, zumindest die Verwendung des Präsentationsdienstes ermöglichen. Beim System OVID ist dies nach dem derzeitigen Stand der Dinge jedoch nicht gegeben, da es zur Präsentation von Lehrinhalten in Form von HTML-Seiten mit integrierten Audio/Video-Informationen eine Netzanbindung mit einer Bandbreite von mindestens 1.2 Mbit/s voraussetzt.

Kapitel 6

Zusammenfassung und Ausblick

Das im Rahmen dieser Arbeit entwickelte Distance-Learning-System ist für den Einsatz unter Windows 98/Me/NT/2000/XP™geeignet. Es basiert auf der Client/Server-Architektur und stellt in seiner Haupteigenschaft dem Nutzer die aufbereiteten Lehrinhalte mit Hilfe eines Web-Servers in Form von HTML-Seiten über das Internet zur Verfügung, welche zur Unterstützung des Lernprozesses unter anderem auch Audio/Video-Informationen beinhalten. Die grundlegenden Eigenschaften des hier betrachteten Systems werden nachfolgend aufgeführt:

- Unterstützung von schmalbandigen und breitbandigen Netzwerken
- leicht zu bedienendes Benutzerinterface
- Werkzeuge für die Gruppenarbeit
- automatische Ressourcenkontrolle
- einheitliches Sitzungsmanagement für alle integrierten Dienste
- modulare Struktur.

Zum Zugriff auf die Lehrinhalte und zur Nutzung der Zusatzdienste wird dem Anwender ein Client-Modul zur Verfügung gestellt, welches alle von den einzelnen Diensten bereitgestellten Funktionalitäten zentral in sich vereint. Die Anzeige der Vorlesungsinhalte erfolgt dabei über den im Client-Programm integrierten Web-Browser, welcher auf dem Browser-Kern des Microsoft Internet Explorers™basiert und mit der auf dem Anwendersystem installierten Version dieser Software-Komponente automatisch aktualisiert wird. Die Audio/Video-Inhalte, welche in den HTML-Seiten der Vorlesungsinhalte integriert sein können, sind sowohl über schmalbandige als auch über breitbandige Netzanbindungen vom Anwender abrufbar. Der Einsatz von Streaming-Technologie erlaubt die Verwendung eines Streaming-Servers zur Bereitstellung der Audio/Video-Daten, welcher diese Daten in Abhängigkeit von der dem Anwender zur Verfügung stehenden Bandbreite seiner Netzanbindung automatisch in unterschiedlichen

Qualitätsstufen übermittelt. Diese Qualitätsstufen unterscheiden sich in der Bildauflösung, der Bildwiederholrate und der Stärke der Komprimierung für die jeweiligen Audio/Video-Daten, so dass einerseits der Vorlesungsdienst auch für schmalbandige Netzanbindungen zur Verfügung steht, und andererseits bei breitbandigen Netzanbindungen deren Möglichkeiten ausgeschöpft werden. Der Einsatz von Streaming-Technologie ist also im Rahmen dieser Arbeit der Schlüssel zur Erfüllung der Anforderungen seitens der Konzeption des Dienstes zur Präsentation der Vorlesungsinhalte in Bezug auf integrierte Audio/Video-Inhalte, da diese die geforderte automatisierte Skalierung der Audio/Video-Inhalte entsprechend der verfügbaren Bandbreite gewährleistet.

Im Rahmen der Gruppenarbeit, welche den Lernprozess in Form von Lerngemeinschaften unterstützt, werden vom System die Dienste für Chat, Whiteboard und Videokonferenz bereitgestellt, wobei der Videokonferenz-Dienst auch bei Verwendung einer Modemverbindung genutzt werden kann und neben der Gruppenkonferenz über einen Konferenz-Server auch private Konferenzen zwischen zwei Teilnehmern erlaubt. Die Umsetzung dieser in der Konzeption zum Videokonferenz-Dienst festgelegten Anforderungen wird durch die Verwendung des H.323-Protokolls sichergestellt, welches den Austausch audieller und visueller Informationen zwischen den einzelnen Teilnehmern sowie die Kommunikation innerhalb einer Gruppe erlaubt und entsprechend der verfügbaren Bandbreite verschiedene Qualitätsstufen mit entsprechend unterschiedlichem Datenaufkommen bereitstellt. Für Nutzer dieses Dienstes, die über eine Netzanbindung mit einer Bandbreite kleiner als 56 Kbit/s verfügen, steht der Videokonferenz-Dienst eingeschränkt zur Verfügung. Diese Einschränkung beinhaltet, dass keine visuellen Informationen mit diesen Teilnehmern ausgetauscht werden, aber wohl eine Kommunikation in audieller Form gegeben ist.

Der Chat-Dienst als weiteres Werkzeug der Gruppenarbeit erlaubt die Nutzerkommunikation in Form des Austausches von Textnachrichten innerhalb von Chat-Gruppen, wobei die Nachrichten der einzelnen Mitglieder einer Chat-Gruppe entsprechend ergonomischen Gesichtpunkten beim Chat-Nutzer absenderbezogen in unterschiedlicher Schriftfarbe dargestellt werden. Der Whiteboard-Dienst stellt seinerseits eine Reihe von Zeichenwerkzeugen bereit, welche den Anforderungen zur Erstellung komplexer Zeichnungen genügen. Hierbei werden auch die Verwendung von Füllmuster für Zeichnungsobjekte und das Einfügen von Text unterstützt. Der zur Beschreibung einer Zeichenoperation und im Rahmen des Datenaustausches verwendete Datentyp zeichnet sich durch die Besonderheit aus, dass er in Hinsicht auf seine Größe optimiert ist, um das beim Whiteboard-Server-Modul durch die Datenverteilung entstehende Datenaufkommen gering zu halten. Die Synchronisations- und Steuervorgänge für die hier besprochenen Dienste zur Gruppenarbeit laufen automatisch ab.

Mit den Zusatzdiensten für eMail, Kurznachrichten und FTP werden dem Anwender weitere Möglichkeiten zum Austausch von Informationen beziehungsweise Daten bereitgestellt. Während der eMail-Dienst beispielsweise zur Informationsübermittlung an Teilnehmer, wel-

che zum Zeitpunkt dieser Übermittlung nicht am Distance-Learning-System angemeldet sind, benutzt werden kann, dient der Kurznachrichten-Dienst zur Übermittlung von Mitteilungen an einen am System angemeldeten Nutzer, um ihn beispielsweise zum Beitritt in eine Chat-Gruppe aufzufordern. Der FTP-Dienst erschließt seinerseits die Möglichkeit zum Download von Begleitmaterial zum Vorlesungsinhalt beziehungsweise zum Austausch von Dateien zwischen verschiedenen Nutzern.

Zur Speicherung der Nutzerdaten wird eine SQL-fähige Datenbank verwendet, welche im Rahmen dieses Projektes konzipiert und implementiert wurde. Das Server-Modul enthält zudem ein grafisches Benutzerinterface, welches die Einsicht und Manipulation dieser Daten auf einfache Art und Weise ermöglicht.

Die kompakte und übersichtliche Gestaltung der Benutzeroberfläche der Anwenderapplikation gestaltet die Nutzung der einzelnen Dienste komfortabel, so dass auch Anwender, die noch nicht mit diesem System vertraut sind, das Distance-Learning-System bedienen können. Zur Übersicht über die vorhandenen Gruppen und deren Mitglieder sowie die Möglichkeit zum Beitritt in eine solche Gruppe werden dem Anwender hierbei für den jeweiligen Dienst entsprechende Listen bereitgestellt, welche die Gruppennamen und Nutzerkennungen enthalten.

Die Implementierung eines einheitlichen Sitzungsmanagements für den Präsentationsdienst sowie die Zusatzdienste Chat, Whiteboard und Videokonferenz erlaubt die zentrale Darstellung und Protokollierung aller hierbei ablaufenenden Vorgänge durch das Server-Modul. Außerdem erhält der Administrator hierdurch auch die Möglichkeit, in den Ablauf des Geschehens einzugreifen, indem er einen Nutzer beispielsweise bei unangemessenem Verhalten aus einem Dienst entfernt. Die Integration der Dienst-Server-Module für Chat und Whiteboard vereinfacht hierüber hinaus die Umsetzung eines einheitlichen Sitzungsmanagement, da hiermit eine gemeinsame Nutzung des Sitzungsmanagers durch den zentralen Server sowie die hier besprochenen Dienst-Server-Module ermöglicht wird, ohne zusätzliche Kommunikationskanäle zwischen den Dienst-Server-Modulen und dem zentralen Server einrichten zu müssen.

Die einzelnen Dienste des hier vorliegenden Systems beinhalten unterschiedliche Voraussetzungen bezüglich der auf dem Client-Rechner vorhandenen Hardware und Software. Deshalb wird beim Programmstart von der Anwenderapplikation eine automatische Ressourcenabfrage durchgeführt, deren Ergebnis an das Server-Modul weitergeleitet und dort ausgewertet wird. Vom Server-Modul wird dem Client-Programm hieraufhin mitgeteilt, welche Dienste dem Anwender zur Verfügung gestellt werden, und welche nicht.

Zur Bewertung der Leistungsfähigkeit sowie der Fähigkeit der Nutzer- und Gruppenverwaltung wurden im Rahmen dieser Arbeit verschiedene Tests für das hier entwickelte System durchgeführt, welche den Einsatz des Systems auf unterschiedlicher Hardware und bei variierenden Nutzeraktivitäten berücksichtigen. Dabei erwies sich das System als stabil. Die hierbei ermittelten Messwerte bezüglich der auftretenden Prozessorlast beziehungsweise Netzwerklast an den unterschiedlichen Rechnersystemen auf Client-Seite zeigen, dass die Nutzung des

Dienstes zur Präsentation der Vorlesungsinhalte mit integrierten Audio/Video-Inhalten sowie die Nutzung des Videokonfernz-Dienstes Prozessorlasten verursacht, welche beim verwendeten Testrechner mit einer CPU vom Typ Pentium I (166 MHz) und 32 MB Arbeitsspeicher die Nutzung dieser Dienste infolge von Überlastungen der CPU verhindern. Die Software-seitige Kodierung und Dekodierung der Videoinformationen ist hierfür verantwortlich. Die beiden leistungsfähigeren Testrechner mit Prozessoren des Typs Pentium III (1200 MHz) und AMD Athlon (800 MHz) und jeweils 512 MB Arbeitsspeicher hingegen bewältigten die Ausführung dieser Dienste problemlos. Alle übrigen Dienste des GWOTS verursachen nur geringe Prozessorlasten von maximal 14% in Abhängigkeit vom Testrechner. Die ermittelten Netzwerklasten auf Client-Seite zeigen, dass die Dienste des Systems GWOTS auch bei Verwendung einer Modemverbindung in vollem Umfang bereitgestellt werden. Weitere Einzelheiten zur Bewertung der Leistungsfähigkeit sind im *Kapitel 5.1* nachzulesen. Um eine endgültige Bewertung für das entworfene System abgeben zu können, muss es jedoch anstatt der durchgeführten Simulation noch im realen Lehrbetrieb getestet werden. Erst ein solcher Test wird Aufschluss darüber geben, inwiefern das System Akzeptanz bei den Nutzern findet und welche Schwachstellen es vielleicht in sich birgt.

Ein weiterführender Ausbau des hier entwickelten Systems sieht unter dem Aspekt der Sicherheit die Verschlüsselung von Login und Passwort für die Übertragung sowie die Speicherung dieser Daten in der Datenbank vor. Aufgrund des modularen Aufbaus des Systems ist unter Verwendung der dafür vorgesehen Schnittstellen im Hinblick auf eine Erweiterung des Funktionsumfangs außerdem auch das Einbinden weiterer Komponenten möglich, welche der Unterstützung des Lernprozesses behilflich sind. Auch stehen beispielsweise weitere Konzepte und Untersuchungen in Hinblick auf die Vermittlung von Wissen mittels der digital aufbereiteten Lehrinhalte aus, welche die Abfrage des erlernten Wissens vom Nutzer sowie deren Evaluation beinhalten.

Abkürzungsverzeichnis

API	**A**pplication **P**rogramming **I**nterface
BDE	**B**orland **D**atabase **E**ngine
$Codec$	**Co**der-**Dec**oder
CPU	**C**entral **P**rocessing **U**nit
DLL	**D**ynamic **L**ink **L**ibrary
FTP	**F**ile **T**ransfer **P**rotocol
$GWOTS$	**G**roup **W**ork **O**nline **T**eaching **S**ystem
$HTML$	**H**yper**T**ext **M**arkup **L**anguage
$HTTP$	**H**yper**T**ext **T**ransfer **P**rotocol
IP	**I**nternet **P**rotocol
$ISDN$	**I**ntegrated **S**ervices **D**igital **N**etwork
ITU	**I**nternational **T**elecommunication **U**nion
$JaTeK$	**Ja**va Based **Te**leTeaching **K**it
$JPEG$	**J**oint **P**hotographic **E**xperts **G**roup
LAN	**L**ocal **A**rea **N**etwork
MCU	**M**ultipoint **C**ontrol **U**nit
$Modem$	**Mo**dulator-**Dem**odulator
$MPEG$	**M**otion **P**icture **E**xpert **G**roup
$OVID$	**O**nline-**Vi**deo/Audio-**D**istance-Learning-System
$QCIF$	**Q**uarter **C**ommon **I**ntermediate **F**ormat
RDM	**R**elational **D**atabase **M**odel
$SOHO$	**S**mall **O**ffice **H**ome **O**ffice
SQL	**S**imply **Q**uery **L**anguage
$T-DSL$	**T**elekom **D**igital **S**ubscriber **L**ine
TCP	**T**ransmisson **C**ontrol **P**rotocol
URL	**U**niform **R**esource **L**ocator
VCL	**V**isual **C**omponent **L**ibrary
WWW	**W**orld **W**ide **W**eb

Literaturverzeichnis

[BOR01] *Borland Delphi 6 Entwicklerhandbuch*, Borland Software Corporation, 2001.

[GUL98] Gulich, A. M.: *Videokonferenz-Technik*, Franzis´ Verlag GmbH, 1998.

[HOE02] Hönemann, Matthias: *System-Ressourcen-Erkennung unter Windows*, Diplomarbeit, HTWK Leipzig (FH), Lehrbereich Informatik, Oktober 2002.

[KIV03] *13. Fachtagung Kommunikation in Verteilten Systemen, KiVS 2003*, http://kivs03.uni-leipzig.de, 28.02.2003
http://www.winfoline.de, 28.02.2003.

[KUE01] Künkel, Tobias: *Streaming Media*, Addison-Wesley Verlag, 2001.

[KUR00] Kurkov, Igor: *Sitzungsmanagement und Ressourcenverwaltung für das Distance-Learning-Tool OVID*, Diplomarbeit, Universität Leipzig, Institut für Informatik, Juli 2000.

[FRI99] Friedrich, Prof. Dr. Leonhard: *Teleteaching - eine neue Komponente in der universitären Lehre*, Dr. Josef Raabe Verlags-GmbH, Stuttgart, 1999.

[MUE02] Münz, Stefan: *HTML & Web-Publishing Handbuch*, Franzis Verlag, Poing bei München, 2002.

[MIC02] *Understanding the H.323 Standard*, Microsoft Corporation, 2002, http://www.microsoft.com/windows/NetMeeting/Corp/ResKit/Chapter11/default.asp.

[MIC03-1] *9 Series Codecs - Video*, Microsoft Corporation, 2003, http://www.microsoft.com/windows/windowsmedia/9series/codecs/video.aspx, 03.02.2003.

[MIC03-2] *Best Audio and Video*, Microsoft Corporation, 2003, http://www.microsoft.com/windows/windowsmedia/9series/player/quality.aspx, 03.02.2003.

[OPE03] *Welcome to the OpenH323 Project*, Equivalence Pty Ltd, 2003, http://www.openh323.org, 10.12.2002.

[REN00] Renz, S.: *Synchrone und asynchrone Kommunikationsdienste für das Distance-Learning-Tool OVID*, Diplomarbeit, Universität Leipzig, Institut für Informatik, Juli 2000.

[ROS00] Roskin, Alexander: *Benutzer- und Ressourcen/Dokumentenverwaltung für ein Distance-Learning-Tool zur Erstellung und Verwaltung multimedialen Lehr- und Lernmaterials*, Diplomarbeit, Universität Leipzig, Institut für Informatik, Juli 2000.

[STE00] Steinmetz, Ralf: *Multimedia-Technologie: Grundlagen, Komponenten und Systeme*, Springer-Verlag, Mai 2000.

[WME02] *Windows Media Encoder*, Online-Hilfe, Microsoft Corporation, 2002.

Danksagung

Diese Arbeit wurde am Lehrstuhl „Informationssysteme und Multimediatechnologie“ des Lehrbereichs Informatik der Hochschule für Technik, Wirtschaft und Kultur Leipzig (FH) angefertigt.

Herzlich danken möchte ich:

Herrn ***Prof. Dr. K. Hänßgen***
für die Vergabe der Aufgabenstellung, die Betreuung der Arbeit sowie für die schnelle Hilfe bei auftretenden Problemen.

Selbständigkeitserklärung

Ich erkläre hiermit, dass ich die hier vorliegende Arbeit selbständig angefertigt und nur die unter Quellen angegebene Literatur verwendet habe.

Leipzig, 14. April 2003