Universität Leipzig

Fakultät für Mathematik und Informatik

Institut für Informatik

Diplomarbeit

Benutzer- und Ressourcen-/Dokumentenverwaltung für ein Distance-Learning-Tool OVID zur Erstellung und Verwaltung multimedialen Lehr- und Lernmaterials

Verantwortlicher HSL: Prof. Dr. K. Irmscher

Betreuer: Dr. K. Hänßgen

Leipzig, den 11.07.2000

vorgelegt von

Alexander Roskin

Inhaltsverzeichnis

Verzeichnis der Abbildungen

Verzeichnis der Tabellen

1 Einleitung

Bei der Entwicklung der ersten Netzwerke ging es den Unternehmen vor allem um die gemeinsame Nutzung von Hardwareressourcen und um den Informationsaustausch. Die Verbindung von Computern mit Hilfe von Netzwerken hat die Art, wie Computersysteme organisiert werden, grundlegend beeinflußt. Heutzutage werden die Computer mehr und mehr als Kommunikationszentralen zwischen den Menschen benutzt. Millionen Computer sind miteinander über ein heterogenes Netz verbunden – das Internet.

Das Internet wird für verschiedene multimediale Dienste verwendet [STE98]. Sowohl die textuelle und grafische als auch audio-Information lässt sich bei den Bandbreiten ab 128 Kbit/s (entspricht der Übertragungsrate eines ISDN-Adapters mit zwei gebündelten Kanälen) in einer guten Qualität übertragen. Bei der Benutzung von auf ATM basierten Netzwerken wird bei Audio-/Videoübertragung eine TV-ähnliche (PAL-Auflösung) Qualität erreicht. Dabei ermöglicht der Bandbreitenzuwachs in LANs und WANs die Darstellung aufwendiger multimedialer Inhalte. Ein Gebiet, bei dem die hohe Bandbreite sinnvoll ausgenutzt werden kann, ist Distance Learning [KERR98]. Darunter versteht man virtuelle Vorlesungen, an denen man entweder in Echtzeit teilnehmen kann, oder die zuerst aufgenommen, abgespeichert, und dann vom Benutzer aufgerufen werden können. Dabei werden der Vortragende mit einer digitalen Videokamera aufgenommen und die komprimierten Daten an die Teilnehmer der Veranstaltung verschickt, die sich in speziell ausgerüsteten Räumen befinden oder auch von weit entfernten Rechnern aus dem Internet auf die Daten zugreifen. Auch die Gruppenarbeit wird mit Hilfe spezieller Komponenten unterstützt. Im Moment wird im universitären Bereich an mehreren Projekten gearbeitet, die solche Funktionalität bieten, einige Projekte befinden sich bereits in einer langjährigen Entwicklungsphase, wie zum Beispiel das JaTeK-Projekt der Technischen Universität Freiberg und der Technischen Universität Dresden. Aufgrund der Tatsachen, dass sich JaTeK in einem fortgeschrittenen Stadium der Entwicklung befindet und für den Einsatz an den Hochschulen konzipiert wurde, wird es zu einem Konzeptvergleich hinzugezogen.

Das Ziel dieser Arbeit ist es, das Konzept für ein Distance-Learning-Tool auszuarbeiten, das unter Windows 98/NT/2000 läuft und eine modulare Architektur besitzt. Dieses Tool wird als leicht modifizierbar entwickelt, und die gesamte Architektur erlaubt den einfachen Anschluß von weiteren Diensten und Teilmodulen. Ausserdem ist dieses Tool in den Netzen mit einer

niedrigen Bandbreite lauffähig und macht Gebrauch von einer hohen Bandbreite, falls sie zur Verfügung steht. Des weiteren sieht die Aufgabenstellung dieser Arbeit die Konzeption und den Entwurf eines Benutzerinterfaces vor , das auf der Seite des Nutzers zum Einsatz kommt, einer Benutzer-Datenbank, die möglichst anbieterunabhängig mit den gängigen SQL-Datenbanken zusammenarbeitet, und eines Dienstes zum Darstellen von digital aufbereiteten und abgespeicherten Inhalten von Vorlesungen.

Im Laufe der Entwicklung des Distance-Learning-Projektes sind ausserdem in Rahmen von zwei weiteren Arbeiten das Sitzungsmanagement entwickelt [KUR00] und die Anpassung von Diensten zum Anschluß an das Hauptmodul vorgenommen [RENZ00] worden.

Zum Einsatz kommt das Distance-Learning-Tool als erstes an der Universität Leipzig. Hier werden dank der hohen Bandbreite, die im Netz der Universität zur Verfügung steht, und dank dem speziell ausgerüsteten Teleteaching-Labor, alle entsprechend vorgesehenen Dienste des Tools zum Einsatz kommen können. Eine hohe Bandbreite wird z.B. bei der Videoübertragung gefordert. Das Einsatzgebiet ist aber nicht auf ein Intranet beschränkt. Andere Dienste, die ohne die hohen Bitraten auskommen, können auch aus dem Internet gestartet und benutzt werden.

Bei Konzeption und Entwicklung von neuartigen Softwareprodukten spielt die Benutzerakzeptanz eine entscheidende Rolle. Das im Rahmen der vorliegenden Arbeit geplante Tool wird nicht nur für die Durchführung von Informatik-Vorlesungen konzipiert, sondern soll von Studenten von anderen, viel weniger mit der Technik verbundenen Fächern, problemlos genutzt werden können. Deshalb ist das Benutzerinterface einfach und Einsteigerfreundlich gestaltet. Auch deshalb wurde Windows als Entwicklungsplattform ausgewählt, die zusammen mit dem Visual C++ Compiler von Microsoft die Entwicklung von einfach und intuitiv zu bedienenden Programmen ermöglicht.

Bei der Durchführung von virtuellen Lehrveranstaltungen oder beim Zugriff auf die abgespeicherten Lehrmaterialien soll auch eine Möglichkeit der Kommunikation zwischen den Benutzern des Systems zur Verfügung stehen. Zu diesem Zweck sind im Projekt verschiedene, sowohl synchrone als auch asynchrone Kommunikationsdienste vorgesehen, wie z.B. Chat, WhiteBoard, Videokonferenz usw., auf die im einzelnen im Kapitel 3 eingegangen wird.

Diese Arbeit ist in 6 Kapitel gegliedert. Das Kapitel **1** gibt einen kurzen Überblick über Motivation, Aufgaben und erzielte Ergebnisse der Arbeit. Im Kapitel **2** werden die Basisarchitektur und die Mechanismen erklärt, die in dieser Arbeit zum Einsatz kamen. Das Konzept des Systems wird detailliert im Kapitel **3** beschrieben. Im Kapitel **4** werden die Details der Implementierung erklärt. Im Kapitel **5** erfolgt eine kurze Vorstellung eines an der TU Bergakademie Freiberg und der Technischen Universität Dresden entwickelten Distance-Learning-Systems. Die Konzeptunterschiede zu dem im Rahmen dieser Arbeit entwickelten System werden erläutert. Ergebnisse, die mit dieser Arbeit erreicht wurden, werden im Kapitel **6** präsentiert. Eine Zusammenfassung und ein Ausblick auf weitergehende Möglichkeiten zur Erweiterung des Systems werden im Kapitel **7** vorgenommen.

Das als Ergebnis dieser Arbeit entwickelte System ermöglicht einen komfortablen Zugriff auf die vorbereiteten und im Internet bereitgestellten Lehrmaterialien und ist dank seiner Architektur aufwärts kompatibel und skalierbar. Es werden die Möglichkeiten der Breitbandnetzwerke für die Audio-/Videowiedergabe in den Lehrinhalten genutzt. Für die Gruppenarbeit stehen mehrere Gruppenarbeit-Werkzeuge zur Auswahl. Eines dieser Werkzeuge, ein Dienst zur Audio-/Videokommunikation, ermöglicht eine komfortable Kommunikation zwischen zwei Benutzern. Alle Synchronisations- und Steuervorgänge laufen automatisch ab. Das heisst, dass das OVID ein integriertes System darstellt, dessen Teile aufeinander abgestimmt sind. Aus diesem Grunde gestaltet sich die Nutzung des Systems sehr einfach, es kann auch von einem ungeschulten Benutzer gesteuert werden.

2 Grundlagen

Das Ziel dieses Kapitels ist es, einen Überblick über die Grundlagen zu geben, auf denen die vorliegende Arbeit beruht. Es werden die Begriffe wie das Cooperative Computing, Gruppenkommunikation, Client/Server-Architektur, WWW und HTML usw. erklärt und eingeordnet. In den Kapiteln 3 und 4 werden basierend auf diesen Begriffen die Konzeption und die Implementierung des im Rahmen dieser Arbeit entwickelten Distance-Learning-Systems OVID behandelt.

2.1 Gruppenkommunikation

Die rechnerunterstützte Zusammenarbeit von Nutzern und die Bereitstellung und Verwaltung von Diensten für Konferenzen bildet die Basis für eine grosse Gruppe von Multimedia-Anwendungen, wie z.B. Distance-Learning-Systeme [EDMEDIA 96-98]. Der folgende Abschnitt befasst sich mit Aspekten der Gruppenkommunikation.

2.1.1 Cooperative Computing

Eine breit verfügbare Infrastruktur von vernetzten Rechnern mit der Fähigkeit der Verarbeitung von Audio- und Videoströmen eröffnet Anwendern die Möglichkeit, kooperativ zusammenzuarbeiten und dabei sowohl räumliche als auch zeitliche Entfernungen zu überbrücken. Die Einbindung in Netze und die Integration von Multimedia-Komponenten in die Endsysteme schafft damit für die Benutzer eine Arbeitsumgebung für die gemeinschaftliche und kooperative Arbeit mit Computern. Diese Form der Kooperation wird allgemein unter dem Begriff *Computer-Supported Cooperative Work* (CSCW) beschrieben und zusammengefasst [CSCW92-98].

Aktuell existieren in dem CSCW-Bereich eine Reihe von Anwendungen, wie z.B. E-Mail, News-Gruppen, Applikationen, die den gemeinsamen Zugriff auf Anwendungen(Screen Sharing) erlauben (z.B. ShowMe von SunSoft, die Tcl/Expect-basierte Anwendung kibiz), textbasierte Konferenzsysteme (z.B. Internet Relay Chat IRC, entsprechende Foren bei CompuServe, AOL), Applikationen für Telefonkonferenzen oder Konferenzräume (z.B. VideoWindow von Bellcore), Videokonfernzsysteme (z.B. die MBone-Anwendungen vic, nv

und vat). Des weiteren wurde eine Vielzahl von Systemen implementiert, die einzelne dieser Teilkomponenten zusammenfassen und diese innerhalb einer einheitlichen Nutzeroberfläche präsentieren. Zu diesen zählen seit längerer Zeit u.a. Rapport von AT&T und Mermaid von NEC. Neuere Entwicklungen wie das *Cooperative Document Repository* (CDR) [TBR98] integrieren die Aspekte der kooperativen Dokumentenverwaltung und –bearbeitung mit modernen Technologien der Multimedia-Kommunikation und ermöglichen den Benutzern einen leichten Zugang zu den eingesetzten Technologien und Werkzeugen [STE98].

Dieser Abschnitt beschreibt eine Rahmenarchitektur für das „Cooperative Computing“ und untersucht allgemein gültige Aspekte der Thematik, die an verschiedenen Systemen und Werkzeugen exemplarisch verdeutlicht werden.

2.1.2 Dimensionen der Zusammenarbeit

Die durch Computer unterstützte Zusammenarbeit und entsprechende Anwendungen werden in der Literatur häufig in einem tabellarischen Schema, wie in Tab. 2.1 beschrieben, klassifiziert. Dabei ist die Zuordnung entlang der Bereichsgrenzen teilweise fliessend; so kann z.B. eine Anwendung zur gemeinsamen Erstellung von Software durchaus von Benutzern, die sich am gleichen Ort befinden, verwendet werden. Die Nutzung aufgezeichneter Videokonferenzen zu unterschiedlichen Zeitpunkten ist ebenfalls denkbar [WSM91].

Ort/Zeit	Gleiche Zeit	Unterschiedliche Zeit
Gleicher Ort	Gemeinsame („face-to-face“) Arbeit mit gemeinsam genutzten Anwendungen („Shared Applications“)	Nutzung von gemeinschaftlich genutzten Kooperations-,Planungs- und Entscheidungswerkzeugen
Verschiedener Ort	Videokonferenzen, Verteilte kooperative Dokumentbearbeitung oder Software-Entwicklung	E-Mail, Newsgruppen

Tabelle 2.1: Klassifizierung der durch Computer unterstützten Zusammenarbeit

Zeit

Betrachtet man das Zusammenwirken hinsichtlich des Parameters Zeit, so gibt es mit der synchronen und der asynchronen Kooperation zwei unterschiedliche Modi. Der asynchrone Modus fasst Formen zusammen, bei denen die Zusammenarbeit zeitlich unabhängig und nicht notwendigerweise zum gleichen Zeitpunkt stattfindet, während die synchrone Zusammenarbeit gleichzeitig erfolgt.

Benutzergruppe

Durch die ‚Benutzergruppe' wird beschrieben, ob ein einzelner Nutzer mit einem weiteren zusammenarbeitet, oder ob sich die Kooperation auf eine Gruppe von mehr als zwei Teilnehmern bezieht. Dabei können Gruppen weitergehend wie folgt klassifiziert werden [STÜ94]:

- Eine Gruppe kann während der Zeit ihres Bestehens statisch oder dynamisch sein. Eine Gruppe ist statisch, falls ihre Mitglieder vorherbestimmt sind, und sich die Teilnehmerschaft während einer Aktivität nicht verändert. Falls sich die Teilnehmerschaft während der kooperativ ausgeführten Tätigkeit verändert und neue Teilnehmer zu jedem Zeitpunkt zur Gruppe hinzukommen oder diese verlassen können, ist eine Gruppe dynamisch.
- Bei der Zusammenarbeit können einzelne Nutzer innerhalb der Gruppe unterschiedliche Rollen einnehmen, so können sie als Mitglied der Gruppe (falls sie in einer Beschreibung der Gruppe aufgeführt sind), als Teilnehmer einer Aktivität der Gruppe (falls es ihnen erfolgreich möglich ist, zu einer solchen hinzuzukommen), als Initiator oder Koordinator einer Sitzung, als Verwalter eines Tokens, das zur Sitzungssteuerung verwendet wird, oder als Beobachter auftreten.
- Gruppen können von Mitgliedern gebildet werden, die homogene oder heterogene Eigenschaften und Bedürfnisse hinsichtlich ihrer Einbindung in die kollaborative Umgebung besitzen.

Steuerung

Die Steuerung während der Zusammenarbeit kann sowohl zentralisiert als auch dezentral erfolgen. Bei einer zentralisierten Steuerung existiert ein Koordinator, der eine zentrale Kontrollfunktion ausübt und an den sich jeder einzelne Teilnehmer mit Anforderungen,

Anfragen oder Berichten wendet bzw. von welchem er Aktivitäten zugeordnet und angewiesen erhält. Dagegen führt bei einer verteilten Steuerung jeder Teilnehmer Aktivitäten unter eigener Regie und Kontrolle durch; die Zusammenarbeit wird dabei durch verteilte Kontrollprotokolle organisiert, die ein konsistentes Zusammenwirken gewährleisten.

Andere mögliche Parameter, nach denen die Art der Zusammenarbeit eingeteilt werden kann, umfassen die örtliche Verteilung (Lokalität) und den Grad der expliziten Wahrnehmung der Zusammenarbeit in einem kollaborativen System (Collaboration Awareness).

Die Zusammenarbeit kann sowohl an einem gemeinsamen Platz (z.B. in einer Versammlung einer Gruppe in einem Büro oder Konferenzraum) als auch zwischen Benutzern, die sich an verschiedenen Orten befinden und die mittels eines Kommunikationssystems zusammenwirken, erfolgen. Diese Form wird auch als Telekooperation bezeichnet. Hierbei wird die folgende Definition verwendet: Telekooperation bezeichnet die mediengestützte arbeitsteilige Leistungserstellung von individuellen Aufgabenträgern, Organisationseinheiten und Organisationen, die über mehrere Standorte verteilt sind.

Je nach Ausprägung der Kollaborationskomponente unterscheidet man Systeme, bei denen die Zusammenarbeit für den Nutzer transparent erfolgt (Collaboration-transparent) und solche, bei denen er sich dieser explizit bewusst wird (Collaboration-aware).

Applikationen zur transparenten Zusammenarbeit entstehen z.B. durch Erweiterung existierender Anwendungen (z.B. Textverarbeitungs- oder Tabellenkalkulationsprogramme) um eine Kooperationskomponente, die zunächst unabhängig von einem einzelnen Anwender genutzt wurde. So ermöglichen einige Systeme zur Dokumentenverarbeitung eine transparente Zusammenarbeit, da Editoren, die für Einzelbenutzer vorgesehen waren, für die gemeinsame und gleichzeitige Bearbeitung von Dokumenten, die zwischen mehreren Nutzern geteilt werden, erweitert wurden.

Dagegen wird Software, die speziell für Konferenz- und Kollaboratiosanwendungen entwickelt wurde, wie z.B. ein Videokonferenzsystem, als Collaboration-aware klassifiziert.

Solche Systeme können weiterhin in Computer-augmented Collaborative Systems, bei denen der Aspekt der Zusammenarbeit betont wird, und Collaboration-augmented Computing

Systems, bei denen das Hauptaugenmerk auf dem Aspekt der Verarbeitung durch Computer liegt, unterschieden werden [MR94]. Die computergestützte Zusammenarbeit betont die Unterstützung einer sozialen Aktivität, wie z.B. einer Diskussion oder das Finden und Treffen einer Entscheidung durch den Einsatz von Rechnern und Netzen. Dagegen steht beim Collaboration-augmented Computing eher der Applikationsgedanke, also die Bereitstellung von Applikationen, die den Anforderungen mehrerer gleichzeitiger Nutzer werden, im Mittelpunkt.

2.1.3 Architektur eines Collaborative-Computing-Systems

Gruppenkommunikation umfasst die synchrone oder asynchrone Kommunikation mehrerer Nutzer unter zentraler oder verteilter Steuerung (Abb. 2.1). Ein Architekturmodell für diese Kommunikation umfasst u.A. ein Support-Modell [WSM91].

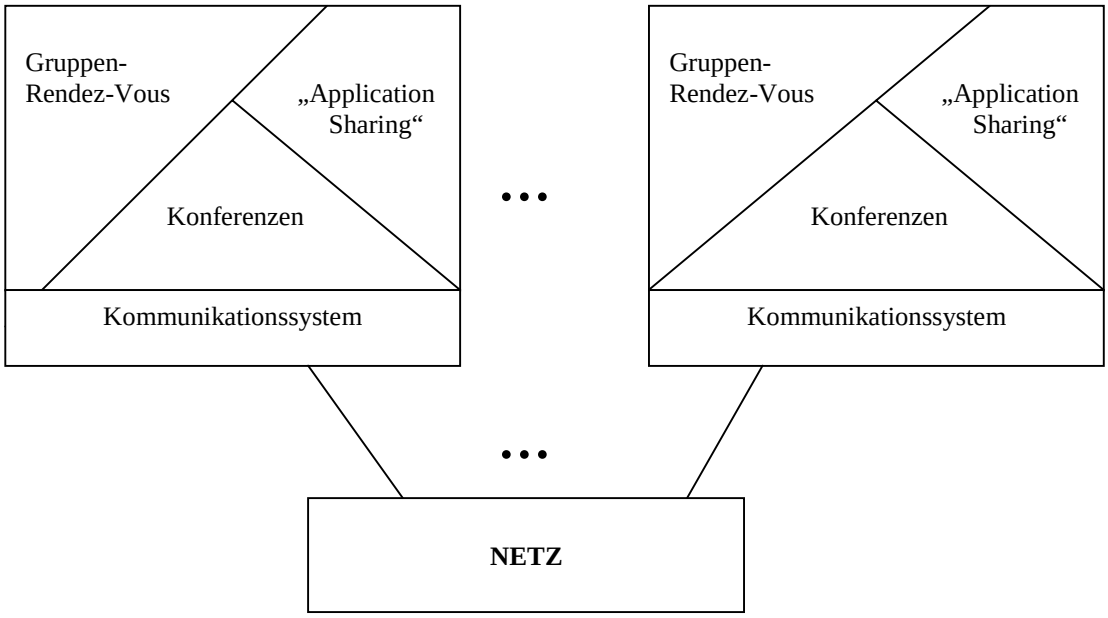

Abb. 2.1: Darstellung der Kooperation von Gruppenkommunikations-Agenten

Das Support-Modell schliesst Gruppenkommunikations-Agenten, die über ein Netz (Abb. 2.1) miteinander kommunizieren, ein. Diese Agenten beinhalten einzelne Komponenten, die speziellen Aspekten gewidmet sind:

Rendez-Vous-Komponente

Dieser Bereich beschreibt Verfahren, die es erlauben, das Zusammentreffen der teilnehmenden Kommunikationspartner zu organisieren (Group Rendez-Vous). Dabei sind statische und dynamische Informationen über die potentiellen oder augenblinklichen Teilnehmer sowie die Spezifika laufender oder zukünftiger Sitzungen zu verwalten und auszutauschen.

Kooperationskomponente (Application Sharing)

Die gemeinsame Nutzung von Anwendungen beschreibt Techniken, die es erlauben, gleichzeitig Informationen für eine Vielzahl von Teilnehmern zu replizieren. Hierdurch entstehen sog. Shared Applications. In diesen können entfernte Teilnehmer z.B. auf spezielle Aspekte einer Information hinweisen (Tele-Pointing) oder diese so modifizieren, dass alle Teilnehmer gleichzeitig an der resultierenden Veränderung teilhaben können (z.B. bei verteilter Dokumentenbearbeitung).

Konfernzkomponente

Konferenzen bilden eine einfache Form des kollaborativen Arbeitens. Die Konferenzkomponente stellt Dienste zur Verfügung, die es mehreren Teilnehmern ermöglichen, unter Nutzung verschiedener Medienströme zu kommunizieren. Hierbei wird gleichzeitig die Verwaltung dieser Teilnehmer unterstützt.

2.1.4 Aufbau der Kommunikationsbeziehungen (Gruppen-Rendez-Vous)

Methoden zum Aufbau der Kommunikationsbeziehungen ermöglichen es, Sitzungen mehrerer Gruppenteilnehmer aufzubauen und stellen statische und dynamische Informationen über Gruppen sowie laufende und zukünftige Sitzungen zur Verfügung. Diese können durch entsprechende Applikationen aufbereitet werden, z.B. nach der Art der Aktivitäten gruppiert und für den Nutzer im Überblick dargestellt werden.

Für das Gruppen-Rendez-Vous wird zwischen synchronem und asynchronem Vorgehen unterschieden:

Synchrone Methoden

Synchrone Methoden nutzen Verzeichnisdienste (Directory Services) und explizite Einladungen. Verzeichnisdienste (wie z.B. X.500, LDAP) ermöglichen den Zugriff auf spezielle Konferenzinformationen, die in einer entsprechenden Wissensbasis gespeichert sind. Diese Informationen beinhalten z.B. den Namen der Konferenz, registrierte Teilnehmer, autorisierte Benutzer sowie Namen und spezielle Funktionen von Teilnehmern. Beispiele für Konferenzsysteme, die die Verzeichnismethode für das Gruppen-Rendez-Vous nutzen sind:

- Die MBone-Werkzeuge für Session Directories *sd* und *sdr*
- Die *Nameserver Query* der Touring Machine von Bellcore, wobei ein Nameserver als zentrale Abspeicher- und Abfragemöglichkeit für sowohl statische als auch dynamische Informationen dient, wie die Menge der autorisierten Nutzer, die registrierten Klienten und die laufenden Sitzungen [LAB93]. Ein Klient kann Informationen zu allen ihn interessierenden Sitzungen, an denen er sich beteiligen könnte, oder zu Teilnehmern einer speziellen Sitzung, abfragen.
- *Directory Service* von MONET. Bei diesem System werden durch einen Directory Server Verzeichnis- und Registrierungsdienste erbracht. Dem Server steht eine Aufstellung verschiedener Ressourcen wie Nutzer, Rechner und Applikationen innerhalb des Netzes, zur Verfügung. Ein Registrierungsdienst erlaubt das Erfassen von Teilnehmern und Nutzergruppen für Konferenzen [SRB92].

Bei der Methode mit expliziten Einladungen erfolgt die Versendung der Einladungen an potentielle Konferenzteilnehmer entweder Punkt-zu-Punkt oder per Multicast. Dabei ist es problematisch, dass der Initiator einer Konferenz wissen muss, wie und wo er andere Nutzer erreichen kann.

Ein Beispiel für den Aufbau der Kommunikationsbeziehungen mittels expliziter Einladungen bildet das *Session Orchestration Tool mmcc*, das vom ISI (USC Information Science Institute) entwickelt wurde [SW94]. Auch in der im MBone genutzten Protokollsuite Mmusic finden sich mit dem *Session Initiation Protocol* (SIP), dem *Session Description Protocol* (SDP) und dem *Session Announcement Protocol* (SAP) entsprechende Möglichkeiten.

Asynchrone Methoden

Asynchrone Methoden können durch die Nutzung von elektronischer Post oder News-Gruppen sowie durch die Präsentation im World Wide Web (WWW) realisiert werden [IET94]. Es wird z.B. E-Mail als Plattform für das Zusammenführen der Gruppenteilnehmer vorgeschlagen. Diese soll in synchrone Konferenzanwendungen eingebettet werden [BOR92].

Der E-Mail-basierte Mechanismus versendet die Informationen, die zum Aufbau einer Sitzung mehrerer Teilnehmer notwendig sind, im Datenteil einer Nachricht. Dieses Schema baut die Adressierung der Teilnehmer als auch für den Versand der notwendigen Informationen auf die bereits vorhandene und bewährte E-Mail-Infrastruktur auf.

2.1.5 Gemeinsame Nutzung von Anwendungen

Die gemeinschaftliche Nutzung von Anwendungen (Application Sharing) wird als ein zentraler und unverzichtbarer Aspekt für die Unterstützung kooperativen Arbeitens betrachtet.

Gemeinsame Benutzung einer Anwendung bedeutet dabei, dass die resultierenden Veränderungen eines verteilten Objektes (z.B. eines Textes oder Bildes) an alle Teilnehmer verteilt und damit für diese sichtbar werden, wenn innerhalb einer verteilten Anwendung (z.B. einem Editor) Eingaben von einem Anwender ausgeführt werden. Anders gesehen werden beim Application Sharing nur Ausgaben verteilt und Eingaben gemultiplext (Window Sharing/Screen Sharing). Verteilte Objekte werden dabei jeweils in sog. *Shared Windows* angezeigt [HTM92].

Application Sharing wird meist in „Kollaborations-transparenten Systemen“ implementiert, kann aber auch im „Collaboration-aware Modus“ in speziellen Anwendungen realisiert werden. Ein Beispiel für ein Software-Toolkit, das die Entwicklung verteilter Applikationen unterstützt, ist das von Bellcore entwickelte Rendez-Vous-System. Gemeinschaftlich genutzte Anwendungen können als unterstützende Komponente in Telekonferenzen zur gemeinsamen

Bearbeitung von Dokumenten oder zur kooperativen Software-Entwicklung eingesetzt werden.

Ein wichtiger Aspekt der gemeinsamen Nutzung von Anwendungen ist die gemeinsame Steuerung und Kontrolle. Als zentrale Architekturentscheidung bei der Entwicklung dieser Applikationen ist festzulegen, ob diese zentralisiert oder durch Replikation realisiert werden [EDMEDIA].

Zentralisierte Architektur

In einer zentralisierten Architektur wird nur eine einzige Instanz der gemeinsam genutzten Anwendung auf genau einem der beteiligten Systeme ausgeführt. Die Eingaben aller Teilnehmer werden zu dieser Anwendung weitergeleitet und die Ausgaben wiederum an alle beteiligten Systeme verteilt. Abb. 2.2 zeigt die zentralisierte Architektur.

Der Vorteil des zentralisierten Ansatzes besteht in der einfachen Verwaltung des resultierenden Zustandes, da nur eine einzige Instanz das geteilte Objekt unmittelbar verändern kann. Der Nachteil ist die grosse Netzlast, da die Ausgaben der Anwendung permanent an alle Teilnehmer verteilt werden müssen. Geschieht dies jeweils vollständig und nicht inkrementell, so ist der aus grafischen Ausgaben resultierende Datenstrom sehr hoch. Daher wird ein Netz mit angemessener Bandbreite benötigt.

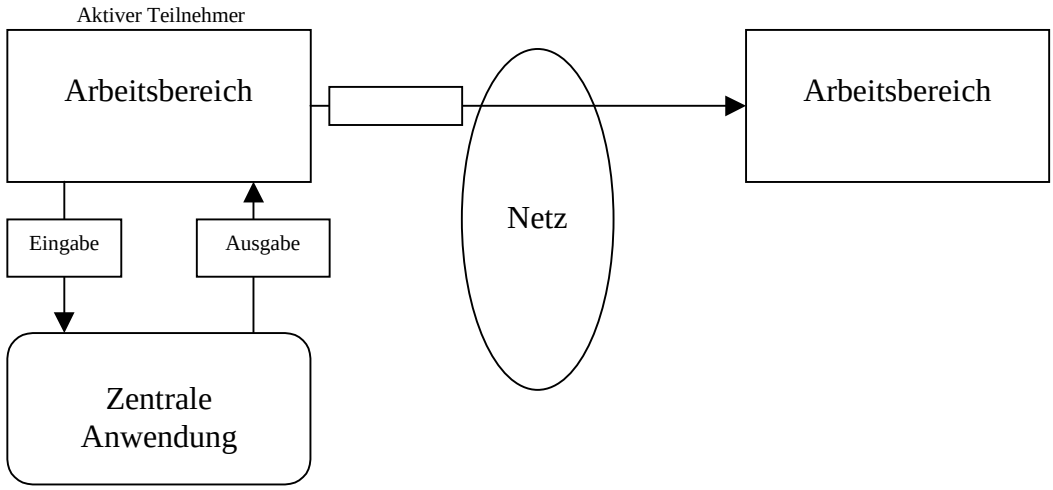

Abb. 2.2: Schematische Darstellung einer zentralisierten Architektur

Architektur mit Replikation

In einer Replikationsarchitektur wird eine Instanz der Anwendung auf jedem der beteiligten Systeme ausgeführt. Eingaben werden jeweils an beteiligte Rechner übertragen und dort lokal verarbeitet. Abb. 2.3 zeigt eine Replikationsarchitektur.

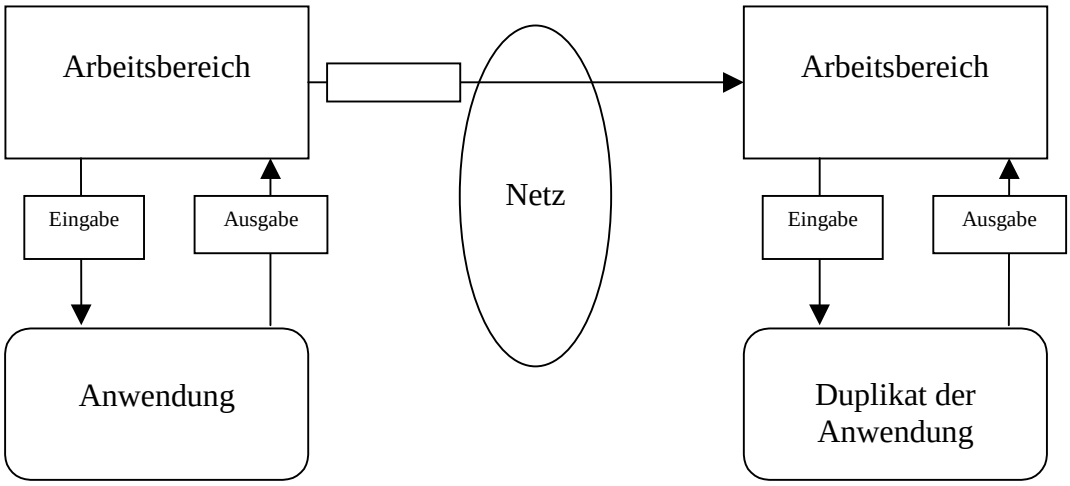

Abb. 2.3: Schematische Darstellung einer replizierten Architektur

Die Vorteile dieses Ansatzes sind:

- Die geringere Netzlast, da nur Eingaben, die in ihrem Umfang in der Regel geringer sind, zwischen den Systemen übertragen werden müssen.
- Die geringeren Antwortzeiten, die sich daraus ergeben, dass die Verarbeitung jeweils lokal erfolgt und die Ausgaben unmittelbar angezeigt werden können.

Von Nachteil ist allerdings die Notwendigkeit einer einheitlichen Ausführungsumgebung auf allen beteiligten Systemen und die erschwerte Aufrechterhaltung der Konsistenz.

Ein wichtiges und schwieriges Problem bei der Umsetzung verteilter Anwendungen besteht bei einer Replikationsarchitektur in der Aufrechterhaltung der systemübergreifenden Konsistenz der gemeinsam manipulierten Objekte. Es existiert eine Vielzahl von Mechanismen, um innerhalb der Menge der Gruppenmitglieder die Datenkonsistenz sicherzustellen. Beispiele dafür sind zentrale Sperrmechanismen (Locks), Mechanismen zur

Übergabe der Aktivität (Floor Passing) und die Bestimmung von Abhängigkeiten (Dependency Detection). An dieser Stelle wird das Verfahren Floor Passing beschrieben, da diese Art der Steuerung für die Gruppenkommunikation die grösste Bedeutung besitzt.

Die Abstraktion einer Staffel (Floor) wird benutzt, um die Konsistenz verteilter Datenobjekte (z.B. Multimedia-Dokumente) oder Applikationen, die gemeinsam von mehreren Anwendern genutzt werden, zu gewährleisten. Nur ein Mitglied der Gruppe, das die Staffel (Floor) aktuell besitzt (Floor Holder) hat das Recht, verteilte Objekte innerhalb des gemeinsamen Arbeitsbereiches (Shared Workspace) zu manipulieren. Folglich kann der Floor Holder Operationen wie das Öffnen oder Schliessen von Arbeitsbereichen sowie das Laden von Dokumenten in Arbeitsbereiche vornehmen. Nur er kann Eingabeereignisse für die gemeinsam genutzten Anwendungen erzeugen und damit die Daten verändern, die allen Nutzern zur Verfügung stehen [STE98], [CSCW92-98].

Eine mögliche Architektur für den Zugriff mehrerer Anwendungen auf gemeinsam genutzte Daten wird in Abb. 2.4 dargestellt.

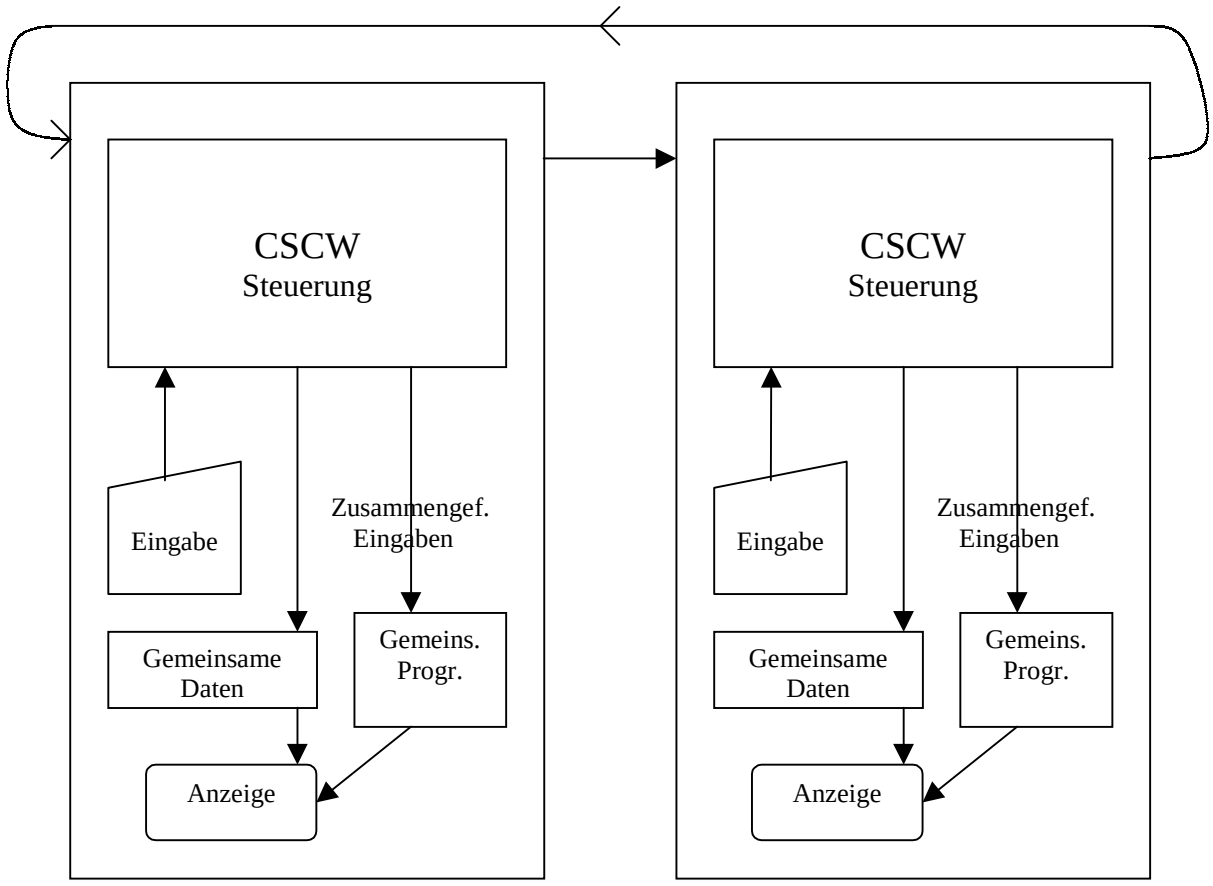

Abb. 2.4: Schematische Darstellung der Architektur für die Manipulation der gemeinsam genutzten Daten

Eine Kontrollkomponente existiert auf jedem der beteiligten Systeme und verteilt die dort von einem Eingabegerät (z.B. Tastatur, Maus) empfangenen Eingaben. Sie überprüft, ob das System momentan zur Menge der Floor Holder gehört. Ist dies der Fall, so werden die Eingaben lokal akzeptiert und verarbeitet und zu den anderen Systemen weitergeleitet. Ist das lokale System kein Floor Holder, dann werden die lokalen Eingaben durch die Kontrollkomponente verworfen. Diese akzeptiert jedoch Eingabeereignisse, die von anderen Systemen übermittelt werden.

Eine Architektur mit Replikation wird in CSCW-Applikationen wie z.B. MERMAID (Multimedia Environment for Remote Multiple Attendee Interactive Decision-making) von NEC oder dem Breitband-ISDN-basierten Gruppen-Tele-Arbeitsystem von Hitachi [HTM92] verwendet.

2.1.6 Konferenzen

Computergestützte Konferenz-Schaltungen unterstützen das kollaborative Arbeiten. Diese Form der Tätigkeit wird auch als synchrone Telekooperation bezeichnet. Notwendig ist dabei ein Dienst, der die genutzten Medienströme an alle Teilnehmer verteilt. Ziel ist es, eine unmittelbare simultane Kommunikation von Angesicht zu Angesicht möglichst unverfälscht nachzubilden. Die dazu eingesetzten Medien Audio und Video dienen in einem Telekonferenzsystem den im folgenden beschriebenen Zwecken.

Videodarstellung

Video wird in technischen Diskussionen benutzt, um die anwesenden Personen direkt oder mit einem Hinweis auf ihre Teilnahme an der Sitzung (z.B. als Bild oder Animation) sowie auch Grafiken, die Teil der Präsentation sind, darzustellen. Gestik und Mimik ist informationstragend; ihre Übertragung schafft eine grössere Ähnlichkeit zur unmittelbaren Kommunikation und ist daher sehr nützlich. Zur Anzeige können Workstations, PCs oder Videowände genutzt werden.

Für Konferenzen mit mehr als drei oder vier Teilnehmer ergibt sich aus dem Platzbedarf zur Anzeige, der für die gleichzeitige Darstellung notwendig ist, schnell ein Problem, insbesondere dann, wenn auch noch weiter Anwendungen wie gemeinsam benutzte Editoren oder Zeichenflächen genutzt werden. Daher sollen dann Mechanismen eingesetzt werden, die es erlauben, die Darstellung eines einzelnen Teilnehmers schnell zu vergrössern, zu verkleinern oder aktivitätsgesteuert in den Vordergrund zu bringen.

Systeme mit Unterstützung für Konferenzräume mit Videowänden versuchen, die gesamte Reichhaltigkeit menschlicher Kommunikation zu unterstützen. Sie erlauben z.B. Seminare mit variablem Teilnehmerkreis, Podiumsdiskussionen oder Lehrveranstaltungen und erschliessen einen grossen Teilnehmerkreis.

Audiowiedergabe

Audio spielt für die Beschreibung und Erklärung der visuell dargestellten Information in einer Konferenz eine sehr wichtige Rolle. Oft ist Audio selbst wichtiger als Video. Deshalb ist eine qualitativ hochwertige Audioübertragung mit einer Voll-Duplex-Kommunikation und Echounterdrückung erstrebenswert. Können zusätzlich noch räumliche Informationen (Stereo) übertragen werden, oder kann eine Zuordnung des aktiven Sprechers zur zugehörigen Bildinformation erfolgen, so kann die Nutzbarkeit weiter verbessert werden.

Konferenzanwendungen verlangen eine möglichst geringe Verzögerung des unterliegenden Netzes sowie eine hohe verfügbare Bandbreite für die teilweise datenintensive Medienübertragung, damit eine akzeptable Interaktivität zwischen den Anwendern möglich ist. Ausserdem benötigen sie einen Dienst zur Verteilung von Nachrichten für die Übertragung von Daten und Steuerungsinformationen an alle Teilnehmer (Distributed Messaging).

2.1.7 Konferenzsteuerung

Konferenzdienste kontrollieren eine Konferenz (d.h. eine Reihe verteilter Informationen, wie den Namen der Konferenz, den Zeitpunkt ihres Beginns, die Menge der Teilnehmer,

einzuhaltende Regeln und weitere Informationen). Die Konferenzsteuerung umfasst verschiedene Funktionen:

- Sie beinhaltet den Aufbau der Konferenz, bei dem sich die Teilnehmer über einen gemeinsamen Zustand, z.B. die Vergabe der Rolle eines Moderators, Zugriffsrechte und –modalitäten (Floor Control) und Koordinierungsvorschriften für die genutzten Medien abstimmen und einigen müssen. Konferenzsysteme können die Registrierung von Teilnehmern, die Zugangskontrolle und das Aushandeln der beschriebenen Parameter bereits während der Aufbauphase der Konferenz durchführen, sollen aber flexibel sein und Teilnehmern das nachträgliche Hinzukommen oder das Verlassen von einzelnen Teilübertragungen (z.B. von Medienströmen) oder der gesamten Konferenz gestatten. Die Flexibilität hängt vom verwendeten Steuerungsmodell ab.
- Es werden Mechanismen zum Hinzufügen neuer Teilnehmer oder zum Entfernen von Teilnehmern, die die Konferenz verlassen, benötigt.
- Das Beenden einer Konferenz muss ebenfalls möglich sein.

Die Mechanismen zur Konferenzsteuerung nutzen die Funktionen des Sitzungsmanagements und kooperieren mit diesen.

Die zu einer Konferenz gehörenden Zustandsinformationen können entweder auf einer zentralen Maschine (zentrale Kontrolle) auf der eine zentrale Applikation als speichernde Instanz arbeitet, oder in verteilter Art abgespeichert werden. Das Steuerungsmodell ergibt sich entsprechend dem Ort, an dem die Zustandsinformation abgelegt ist. Folglich kann auch dieses entweder zentralisiert oder verteilt sein.

Zentralisierte Steuerung einer Konferenz

Bei der zentralisierten Konferenzsteuerung wird die Konferenz zentral aufgebaut. Ein Initiator beginnt die Konferenz, indem er eine anfängliche Gruppe von Teilnehmern auswählt und diese explizit einlädt. Dies setzt voraus, dass er die Adressen aller Konferenzteilnehmer kennt. Das Wissen über den aktuellen Zustand der Konferenz wird von einem zentralen Directory-Server abgefragt, bei dem die Klienten zunächst Informationen über den Aufenthaltsort hinterlegen müssen.

Im zweiten Schritt antwortet jeder eingeladene Klient auf die Einladung, so dass der Initiator darüber informiert ist, wer an der Konferenz teilnimmt. Anschliessend erfolgt die Aushandlung der Regeln für die Konferenz und die Zuteilung und Freigabe von Ressourcen zwischen den Teilnehmern. Während der Aushandlung wird der gemeinsame Konferenzzustand mittels eines Protokolls zum zuverlässigen Versand von Nachrichten an alle Teilnehmer verteilt. Die gesamten Informationen, die die Konferenz betreffen, werden auf einem zentralen System abgespeichert.

Diese statische Steuerung, die durch den expliziten Austausch von Statusinformationen realisiert wird, garantiert für jeden Teilnehmer Konsistenz und ist für kleinere Konferenzen gut geeignet. Die garantierte Konsistenz ist der Hauptvorteil der zentralisierten Steuerung. Ihr Nachteil besteht darin, dass, wenn ein neuer, zunächst nicht eingeladener Teilnehmer zur Konferenz hinzukommen möchte, alle Teilnehmer durch den expliziten Austausch von Informationen über den veränderten Zustand informiert werden müssen, was zu grösseren Verzögerungen führen kann. Weiterhin ist es im Falle der Unterbrechung der Verbindung zu einem Teilnehmer schwierig, den Zustand wieder konsistent aufzubauen.

Verteilte Konferenzsteuerung

Die verteilte Konferenzsteuerung basiert auf einem verteilten Konferenzzustand. Dieser wird wie folgt erreicht: Der Initiator der Konferenz wählt für die Übertragung der Informationen zu den Teilnehmern eine oder mehrere Multicast-Adressen aus, und die Konferenz wird eröffnet. Konfernzteilnehmer können hinzukommen, indem sie den Empfang der speziellen Multicast-Daten aktivieren. Die dazu notwendigen Ankündigungsinformationen (Multicast-Adresse, Port) werden ihnen vorher unter Nutzung der bereits beschriebenen Protokolle für das Gruppen-Rendez-Vous übertragen oder zugänglich gemacht [CSCW92-98].

Jedes beteiligte System überträgt seinen eigenen Teilnahmestatus an die anderen Konferenzteilnehmer. Es gibt jedoch kein zentral verwaltetes Wissen über den Teilnehmerkreis und keine Garantie dafür, dass alle Teilnehmer dieselbe einheitliche Vorstellung vom aktuellen Gesamtzustand besitzen. Nachfolgend wird diese lose Steuerung durch die periodische erneute Übertragung von Zustandsinformationen mit einem ungesicherten Dienst (z.B. sichert IP-Multicast dem Absender weder die Auslieferung an die beabsichtigten Adressaten noch die Unversehrtheit der Daten zu) realisiert. Ziel ist es,

letztlich eine konsistente Sicht bei allen Teilnehmern zu erreichen, auch wenn diese nicht garantiert wird. Diese lose Steuerung ist für grosse Konferenzen gut geeignet. Sie wird z.B. in sog. *Lightweight Sessions* [EDMEDIA], wie sie für Konferenzen im MBone zum Einsatz kommen, angewandt.

Vorteile der verteilten Konferenzsteuerung sind die Fehlertoleranz und die Möglichkeit zur Skalierung auf eine grosse Anzahl von Teilnehmern. Wenn während einer Konferenz eine Netzverbindung ausfällt und anschliessend wieder verfügbar wird, ist es leichter, den gemeinsamen Konferenzzustand wieder aufzubauen, da es für diesen keine strengen Konsistenzanforderungen gibt.

Nimmt eine grosse Anzahl von Benutzern an der Konferenz teil, was zunächst gut unterstützt wird, so erfolgt eine Anpassung der Sendezeitpunkte und der Intervalle zum Austausch der Zustandsinformationen an die Grösse und Ausdehnung der Konferenz. Anderenfalls besteht die Gefahr, dass es zu einer Überflutung mit entsprechenden Nachrichten kommt. Der Nachteil des verteilten Ansatzes besteht darin, dass nicht gewährleistet ist, dass alle Teilnehmer dieselbe Sicht und Vorstellung vom aktuellen Zustand der Konferenz haben.
Die Versuche, ein allgemeines und generisches Protokoll zur Konferenzsteuerung abzuleiten, wurden bereits z.B. mit dem Conference Control Channel Protocol (CCCP) [STE98] unternommen.

Um einen gemeinsamen Zustand abzustimmen, können State-Agreement-Protokolle genutzt werden. Ein solcher Zustand wird auch als *Ephemeral Teleconferencing State* bezeichnet [STE98]. Der Zustand ist kurzlebig, da er nur für die Dauer der Konferenz gültig und von Bedeutung ist.

Agreement-Protokolle mit verteilter Steuerung schliessen spezielle Regeln (Policies) zur Kontrolle des Zustandes einer Sitzung ein. Für diese Regeln werden drei Aspekte ausgemacht [STE98]:

- Der erste Aspekt, als Initiator-of-Policies bezeichnete, gibt an, welche Teilnehmer bestimmte Änderungsoperationen initiieren dürfen.
- Der unter Voting-Policies zusammengefasste zweite Aspekt beschreibt Regeln für die Entscheidung über die von einem Teilnehmer initiierte Änderung des Zustandes. Die Veränderung des gemeinsamen Zustandes basiert auf Abstimmungsregeln (Voting Rules).

- Der dritte Aspekt betrifft Regeln für die Gewährleistung der Konsistenz (Consistency Policies). Um diese zu erreichen und zu erhalten, sind u.a. die Art der Einigung auf einen gemeinsamen Zustand, die Funktionen zur Floor- und Zugangskontrolle, die Aushandlung der benutzten Medien und deren Kodierung, die Verzeichnisdienste für Konferenzen oder auch die Dienste für die Einladung oder das Auffinden von Teilnehmern zu definieren und festzuschreiben.

2.2 Client/Server Architektur

Da die Gruppenkommunikation des im Rahmen dieser Arbeit entwickelten Distance-Learning-Systems auf die Client/Server Architektur aufsetzt, ist es an dieser Stelle erforderlich, einen Überblick über das Client/Server-Modell zu geben.

Die Entwicklung der Client/Server Architektur hat zum Wechsel der Paradigmen in der Industrie geführt. Die grossen Mainframe-Applikationen wurden durch die Applikationen ersetzt, die miteinander über kürzere oder längere Verbindungsstrecken kommunizierten. Der Client – typischerweise ein PC – stellt dabei eine grafische Benutzeroberfläche zur Verfügung, während der Server Zugriffe auf die geteilten Ressourcen - in der Regel verschiedene Arten der Datenbanken – erlaubt. Die neueste Entwicklung im Client/Server Bereich stellen verteilte Objekte im Internet dar [ORF99].

Obwohl ‚Client/Server' vielleicht das meist genannte Wort neben dem Wort ‚Internet' in der Computerindustrie ist, gibt es keine Absprachen darüber, wie dieser Begriff zu definieren ist. Was macht die Client/Server-Architektur anders als andere Arten von verteilten Systemen ?

In der Literatur werden folgende Charakteristika genannt [ORF99]:

- *Service* – Client/Server ist eine Beziehung zwischen Prozessen, die auf separaten Rechnern laufen. Der Server stellt Services zur Verfügung.
- *Shared Resources* – Ein Server kann gleichzeitig mehrere Anfragen von verschiedenen Clients bearbeiten und reguliert ihren Zugriff auf die geteilten Ressourcen.

- *Asymmetrical Protocols* – Es existiert eine n:1 Beziehung zwischen den Clienten und dem Server. Clients initiieren die Verbindung, indem sie den Server kontaktieren und seine Services beanspruchen. Server laufen passiv und erwarten Anfragen von den Clienten.
- *Transparency Of Location* – Ein Server ist ein Prozess, der entweder auf derselben Maschine wie der Client läuft oder aber auf einer anderen, die über das Netzwerk erreicht werden kann. Ein Programm kann sowohl ein Client als auch ein Server oder auch beides gleichzeitig sein.
- *Mix-And-Match* – Die ideale Client/Server Software sollte möglichst hardware- und betriebsystemunabhängig sein.
- *Message-Based Exchanges* – Clients und Server sind ein schwach gekoppeltes System, das über einen Message-Austauschmechanismus kommuniziert, der die Anfragen und Antworten zustellt.
- *Encapsulation Of Services* – Ein Server ist ein ‚Spezialist'. Eine Nachricht sagt dem Server, was für ein Service angefordert wird. Der Server entscheidet dann, wie diese Anfrage zu bearbeiten ist. Solange das Nachrichtenaustauschinterface unverändert bleibt, kann der Server beliebig oft aufgerüstet oder verändert werden.
- *Scalability* – Client/Server-Systeme können sowohl horizontal als auch vertikal skaliert werden. Horizontale Skalierung bedeutet das Hinzufügen oder das Entfernen von Clientrechnern mit nur geringer Veränderung in der Leistung. Vertikale Skalierung bedeutet entweder eine Umrüstung auf grössere und schnellere Serverrechner oder eine Verteilung der Auslastung auf mehrere Server.
- *Integrity* – Der Code und die Daten des Servers werden zentral gemanaged, was in einer preiswerten Wartung und Gewährleistung der Datenintegrität resultiert. In der selben Zeit bleiben die Clients unabhängig.

Es existieren mehrere Arten von Servern. In den folgenden Abschnitten sind einige wenige aufgelistet, die am häufigsten gebraucht werden und die für die vorliegende Arbeit relevant sind.

2.2.1 Datenbank-Server

Datenbank-Server bearbeiten SQL-Anfragen als Client-Nachrichten. Die Ergebnisse jeder Anfrage werden über das Netzwerk zurückgegeben. Der Code, der die SQL-Anfrage

bearbeitet, und die Daten befinden sich auf ein und derselben Maschine. Der Server benutzt seine eigene Rechenleistung, um die benötigten Daten von den anderen zu trennen, und liefert nur die gefilterten Ergebnisse an den Client zurück. Im Endergebnis wird die Netzbelastung so niedrig wie möglich gehalten. Der Applikationscode befindet sich auf dem Clientrechner.

2.2.2 Transaction-Server

Bei einem *Transaction-Server* werden vom Client die Funktionen, die sich auf dem Server mit der SQL-Datenbank befinden, aufgerufen. Diese Funktionen führen dann eine bestimmte Anzahl von SQL-Anfragen aus. Über das Netzwerk wird nur eine einzige Anfrage/Antwort-Nachricht übertragen (im Unterschied zu dem Datenbank-Server, wo jede einzelne SQL-Anfrage mit einer Nachricht beantwortet wird). Diese Gruppe von SQL-Anfrage wird entweder vollständig erfolgreich ausgeführt, oder im Falle eines Fehlers als Ganzes zurückgewiesen. Eine solche Gruppe von Anfragen wird auch Transaction genannt. Beim Entwurf von einem Transaction-Server wird sowohl für den Client als auch für den Server Programmcode geschrieben. Die Clientkomponenten bestehen dabei aus einem grafischen Benutzerinterface, und die Serverkomponenten stellen SQL-Transaktionen mit der Datenbank dar. Diese Applikationen werden auch OLTP, oder Online Transaction Processing genannt..

2.2.3 File-Server

Bei einem *File-Server* sendet ein Client die Fileanfragen an den Server. Es ist eine primitive Form von Services, die einen regen Nachrichtenaustausch über das Netzwerk notwendig macht, um benötigte Dateien zu finden. Fileserver dienen der Verteilung von Daten über das Netzwerk, z.B. Dokumenten, Bildern usw., die von mehreren Benutzern bearbeitet werden sollen.

2.2.4 Object-Applikation-Server

Bei einem *Object-Application-Server* werden die Applikationen als ein Satz von kommunizierenden Objekten geschrieben. Dabei kommt ein sogenannter ORB (Object Request Broker) zum Einsatz. Der Client ruft bei diesem Ansatz die Funktionen des entfernten Objekts auf. Der ORB lokalisiert eine Instanz der Server-Klasse, ruft die Methode auf und gibt das Ergebnis an das Client-Objekt zurück. Das Server-Objekt muss dabei das s.g. Sharing unterstützen. Heute wird das alles von den ORBs und der neuen Generation von CORBA Application Servers erledigt. Es gibt mehrere kommerzielle ORBs, die den Standard der Object Management Group erfüllen, z.B. Orbix von Iona, VisiBroker von Inprise usw.
CORBA ist auch die fundamentale Technologie für das Enterprise JavaBeans Komponenten Model.

Allerdings ist CORBA nicht der einzige Einsatz, der auf diese Technologie aufsetzt. Microsoft hat seinen eigenen ORB, der Distributed Component Object Model oder kurz DCOM genannt wird. DCOM ist die grundlegende Technologie im ActiveX-Komponenten Modell. Der Microsoft Transaction Server (MTS) ist ein Applikation-Server für die ActiveX-Komponente.

2.2.5 Web-Application-Server

Das World Wide Web ist die erste wirklich ‚intergalaktische' Client/Server-Applikation. Dieses neue Modell in der Client/Server-Welt besteht aus kleinen, portablen Clients, die mit den übergrossen *Web-Application-Servern* kommunizieren. In ihrer einfachsten Form liefern die Web-Server Dokumente zurück, die von Clients angefordert werden. Die Clients kommunizieren mit dem Server über ein Protokol namens HTTP. Dieses Protokol definiert einen einfachen Satz von Befehlen; die Parameter werden als Zeichenketten übergeben.

Das Web-Client/Server-Modell befindet sich zur Zeit in einem Zustand der Entwicklung. Die Web-Technologie verschmilzt zunehmend mit der Technologie der verteilten Objekte zu einer mehr und mehr interaktiv orientierten Form der Client/Server Architektur. Diese neue Form wird ‚Object-Web' genannt. Java-Applets und die Browser mit der CORBA-Unterstützung sind die ersten Manifestationen dieses neuen Object-Webs. Im parallelen Universum von

Microsoft wird dieses Modell durch die ActiveX-Komponenten representiert, die über COM+ oder HTTP kommunizieren.

Funktionell sind die Web Application Server ähnlich zu den Object Servern. In naher Zukunft wird es aber unmöglich sein, zwischen ihnen zu differenzieren. Zum Beispiel ist schon jetzt ein MTS Distributed Object Server in der Microsoft Welt gleichzeitig auch ein Web Application Server und benutzt den HTTP-Server von Microsoft als Front-End.

2.3 World Wide Web und HTML als Formatierungssprache

Bei der Implementierung eines Dienstes zur Wiedergabe von abgespeicherten Vorlesungen kommt HTML mit Erweiterungen wie CGI oder MS-ActiveX (siehe Anhang) als die Sprache zum Einsatz, mit der die multimedialen Dokumente erstellt werden. Die Verwaltung dieser Dokumente übernimmt ein Web-Server. Hier folgt eine kurze Beschreibung des World Wide Webs und der HTML als Formatierungssprache im Web. Auf die hier erklärten Begriffe wird in den Kapiteln 3 und 4 eingegangen.

2.3.1 WWW

Das Web stellt sich nach aussen als eine Vielzahl formatierter Seiten dar, die untereinander durch sogenannte Hyperlinks verknüpft sind. Klickt der Benutzer einen im Dokument gekennzeichneten Hyperlink an, wechselt das System zu der damit verbundenen Seite.

Das Prinzip dieser Navigation ist einfach. Dokumente im World Wide Web (WWW) sind Dateien, die mit HTML (Hypertext Markup Language), einer Textformatierungssprache geschrieben sind. Der Browser lädt eine Datei von einem entfernten Rechner, indem er sich dort ankoppelt, dem Server ein GET-Kommando für eine bestimmte Dataei schickt, diese entgegennimmt und anschliessend entsprechend den Formatierungskommandos ihres Textes anzeigt. Ein Klick des Benutzers auf einen Hyperlink bewirkt, dass die nächste HTML-Datei, spezifiziert durch eine URL (Uniform Resource Locator) geladen und angezeigt wird [KLU96].

Eine URL bezeichnet eindeutig eine bestimmte Datei auf einem beliebigen Rechner des Netzes. Für HTTP, das Protokoll des World Wide Webs, haben sie die Form:

http://remote.host.com/path/file.html

Der Zielrechner der Anfrage wurde hier mit `remote.host.com` angegeben, die gewünschte Datei dort ist `file.html` im Verzeichnis `path.http`. Das Protokoll bestimmt, nach welchen Regeln die Kommunikation zwischen lokalem und Zielrechner abläuft. So muss der lokale Rechner nach dem HTTP-Protokoll lediglich einen GET-Request aussenden, während bei FTP beispielweise eine Anmeldung mit Name und Passwort erforderlich wäre.

Das transferierte Dokument kann man mit einem normalen Ascii-Texteditor betrachten, aber die HTML-Kommandos erschweren die Lesbarkeit. HTML ist eine Textformatierungssprache, deren Kommandos in <>-Klammern gekapselt mitten im Text stehen und festlegen, wie die logische Darstellung des Dokumentes aussehen soll.

2.3.2 HTML

Es gibt mehrere verschiedene Versionen von HTML. HTML wird von einer Arbeitsgruppe namens *Internet Engineering Task Force* (IETF) in Zusammenarbeit mit dem industriellen Konsortium W3C (Kürzel für WWW-Consortium) definiert. Im Dezember 1997 wurde die neue Version des Stadardes – HTML 4.0 – verabschiedet, die auch zum Zeitpunkt des Schreibens dieser Arbeit aktuell bleibt. Diese Version enthält viele neue Elemente wie *Cascading Style Sheets* (CSS), Internationalisierung, und einen neuen Object-Befehl. Ausserdem wurden bei diesem Standard auch viele ältere Mechanismen, wie z.B Tabellen, verbessert. Der Microsoft Internet Explorer ab Version 4.0 basiert auf dem Standard HTML 4.0 [KLU96].

Die Formatierungsanweisungen geben an, ob der beistehende Text als Überschrift gedacht ist, ein neuer Absatz beginnt oder ein Bild eingefügt wird. HTML unterstützt eine strikte Trennung des Textinhalts von seiner Darstellung. HTML definiert dabei sogenannte Tags, die einen Textabschnitt als eigenständige Einheit definieren und ihm Eigenschaften zuweisen. HTML legt dabei nicht exakt fest, wie gross und mit welchem Schriftfont eine Überschrift

gesetzt ist, sondern definiert lediglich, dass ein Textfragment eine Überschrift ist. Das endgültige Layout bestimmt immer der interpretierende Browser.

2.3.3 Communikation zwischen dem Web-Server und den Browsern

Damit sich Web-Server und Client auch verstehen, tauschen sie vor und bei der Dokumentübergabe noch wichtige Informationen aus. Da im Netz unterschiedlichste Partner miteinander kommunizieren müssen, ist es wichtig, einen kleinsten gemeinsamen Nenner der Kommunikation zu finden.

So definiert der Server das Format eines Dokuments durch die standardisierten MIME-(Multipurpose Internet Mail Extensions-)Header. Diese geben unter anderem die Länge des übermittelten Dokuments, die Art seines Inhalts (Video, Audio, Bilddaten, Text) und das verwendete Format (für Text z.B. HTML, Plaintext) an. MIME-Header entstammen der E-Mail-Welt und dienen dort dazu, den Inhalt von Multimedia-Sendungen zu spezifizieren [SCHI97].

Ein Server liefert nach diesem Verfahren vor den gewünschten Daten deren Format und gibt so dem Client die Möglichkeit, sie mit geeigneten Mitteln weiterzuverarbeiten. So zeigt ein Browser eine im Klartext gesendete Seite ohne Umschweife an, während eine HTML-Seite vorher durch den browsereigenen Formatierer läuft.

Nicht nur der Server, der ein Dokument liefert, sendet einen Header, sondern auch der anfragende Client , der diesen Kommunikationsweg nutzt, um dem Server vorab mitzuteilen, in welcher Form dieser die Information – falls möglich – übergeben soll. Eine Client-Anfrage spezifiziert so nicht nur das angeforderte Dokument, sondern sendet dem Server auch noch MIME-codierte Formatwünsche.

Den Inhalt der übermittelten Header-Felder bekommt der Benutzer eines Browsers nicht zu Gesicht. Es kann aber vorkommen, dass eine Seite nicht mehr an der angegebenen Stelle existiert, sondern verlagert wurde. Die Antwort des Servers enthält dann im Status-Feld des Headers einen entsprechenden Vermerk und im Feld ‚Location‘ den neuen Standort des Dokuments. Diese Redirect-Angabe wird von dem Browser verarbeitet, und er versucht

sofort, das Dokument von der neuen Location zu laden. Im Erfolgfall merkt der Benutzer nicht, auf welchen Umwegen sein Request abgearbeitet wurde.

Den Fehler-Status einer Anfrage übermittelt der Server vor den eigentlichen Header-Zeilen. Falls etwas schiefgelaufen ist, enthält die erste Zeile der Server-Antwort eine Fehlernummer und den beschreibenden Fehlertext.

3 Konzeption

In diesem Kapitel wird das Konzept eines Distance-Learning-Tools erläutert, das im Rahmen der vorliegenden Arbeit entwickelt wurde. Es wird erklärt, welche Aspekte bei der Konzeption berücksichtigt wurden und wie das System aufgebaut ist. Die detaillierte Beschreibung des Sitzungsmanagements wird in [KUR00], der einzelnen Dienste in [RENZ00] vorgenommen.

Als erstes ist es notwendig zu spezifizieren, welche Anforderungen das System erfüllt:

- Sowohl die Möglichkeiten der Netzwerke mit einer hohen Bandbreite als auch der Einsatz des Systems in den schmalbändigen Netzwerken werden unterstützt
- Alle angemeldeten Benutzer können das System in Anspruch nehmen. Der Standort des Benutzers und die Ausstattung seines Computers spielen dabei keine Rolle, lediglich ein Internet-Zugang ist erforderlich
- Es werden Werkzeuge für die Gruppenarbeit angeboten
- Teile des Systems sind aufeinander abgestimmt und bieten dem Benutzer ein leicht zu bedienendes Interface
- Das System besitzt eine modulare Struktur, um die Weiterentwicklung einfach zu gestatten, d.h. offene Architektur
- Es ist leicht zu warten

Der Entwurf basiert auf einer Client/Server-Architektur, bei der zwei wichtigste Komponenten zum Einsatz kommen:

1. Benutzerverwaltung und Sitzungsmanagement (Server). Es wird vom Systemadministrator bedient.
2. Alle verfügbaren Dienste (Clients). Sie werden vom Anwender bedient. Nach der Anmeldung am Server stehen alle Dienste zur Verfügung.

Entsprechend der allgemeinen Architektur der Systeme für Gruppenarbeit werden folgende sowohl synchrone als auch asynchrone Dienste bereitgestellt:

1. Wiedergabe von abgespeicherten und digital aufbereiteten Vorlesungsinhalten

2. Chat-Dienst
3. WhiteBoard-Dienst
4. Videokommunikation mit ausgewählten Partnern
5. E-Mail-Kommunikationen
6. FTP
7. Kurzmitteilungen-Austausch-Dienst

Die Wiedergabe von abgespeicherten Vorlesungsinhalten hat bei einem Distance-Learning-Tool eine sehr wichtige Bedeutung. Da alle anderen Dienste Zusatzkomponenten darstellen, spielt sie die zentrale Rolle, d.h. der zentrale Dienst ist für die Wiedergabe von multimedialen Inhalten konzipiert, die Zusatzdienste können auf Wunsch des Benutzers aufgerufen werden.

Einige Dienste werden als Module konzipiert und entworfen, die auch getrennt vom System lauffähig sind, was ihre Weiterverwertung ermöglicht. Sie können aber bei der Integration in das Gesamtsystem vom Sitzungsmanagement kontrolliert werden. Die Abbildung 3.1 zeigt den Aufbau des Systems.

Hier nochmals die Anforderungen, die an einen Dienst gestellt werden:

1. Ein Dienst bietet die Funktionalität an, die für die Gruppenarbeit notwendig ist
2. Es ist ein Teil des Systems, das nahtlos in das gesamte System integriert ist
3. Sobald aktiv, nimmt ein Dienst die Kommunikation mit dem zentralen Server auf, um eventuelle Steuernachrichten zu verschicken oder zu verarbeiten

Dank dem modularen Aufbau des Systems werden die Zusatzdienste dabei als unabhängige Applikationen entwickelt, die dann lediglich um die Funktionalität zur Kommunikation mit dem zentralen Servermodul erweitert werden. Auch bereits existierende Applikationen werden so in das System aufgenommen.

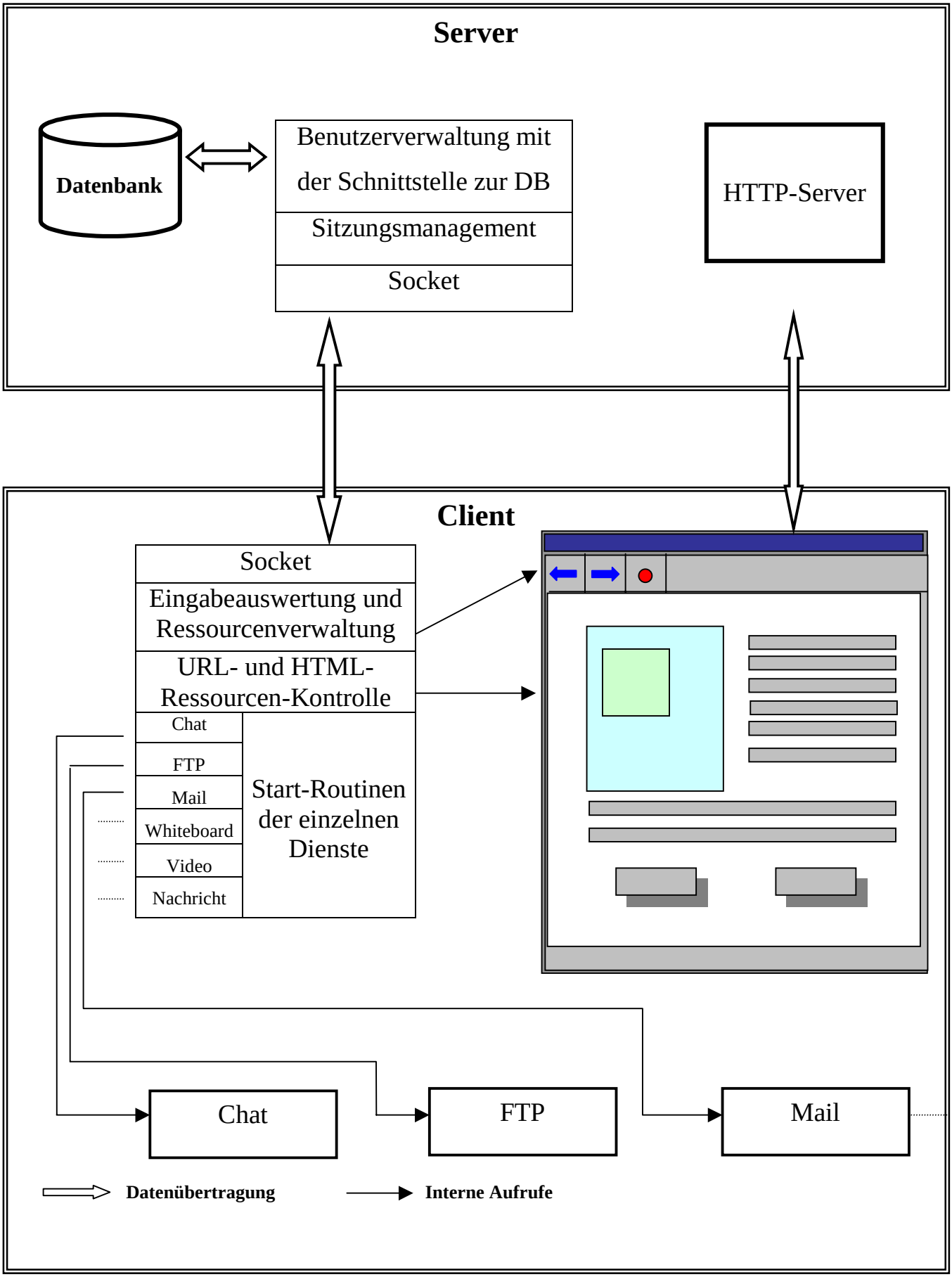

Abb. 3.1: Schematischer Aufbau des Distance-Learning-Systems

3.1 Dokumenten- und Benutzerverwaltung

Beim Konzipieren des Dienstes zur Wiedergabe von Vorlesungen standen zwei Alternativen für die Dokumentenverwaltung zur Auswahl:

1. Die Verwaltung von Inhalten, die aus textuellen, grafischen und audio-visuellen Information bestehen, wird von einer Datenbank durchgeführt, die auch Multimedia-Daten verwalten kann.

   Vorteile dieses Ansatzes:
   - einfache Verwaltung durch Anwendung einer Datenbank
   - hohe Performance

   Nachteile:
   - grosser Implementierungsaufwand für die Darstellung der Inhalte auf der Client-Seite
   - niedrige Flexibilität bedingt dadurch, dass für jeden Datentyp (AVI, MPEG, GIF usw.) eigene Routinen geschrieben werden müssen; Einfügen von neuen Datentypen ist immer mit einem hohen Aufwand verbunden

2. Die zweite Möglichkeit besteht darin, die Vorlesungsdaten in Form von HTML-Seiten abzuspeichern. Dabei verwaltet ein Web-Server die Daten.

   Vorteile:
   - Wiedergabe mit einem Web-Browser möglich
   - hohe Flexibilität dadurch, dass ein Browser alle relevanten Datentype darstellen kann
   - Zukunftssicherheit dadurch, dass die 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

Nachteile:

- Einsatz eines marktüblichen Browsers gefährdet die Integrität des Systems

Die Überlegenheit des zweiten Ansatzes ist aus dieser Auflistung der Vor- und Nachteile klar. Vor allem spielt dabei die Zukunftssicherheit und Erweiterbarkeit eine wichtige Rolle, denn egal, wohin die weitere Entwicklung von WWW und HTML führt, wird das System mit der Installation eines neuen Browsers um die neue Funktonalität automatisch erweitert.

Eine weitere Möglichkeit des Systementwurfs besteht darin, das gesamte Distance-Learning-System als eine webbasierte Applikation anzulegen, d.h. das ganze System läuft dabei unter einem Web-Browser wie dem Internet Explorer von Microsoft oder dem Communicator von Netscape, und nur die von diesen Browsern unterstützten Mechanismen kommen zum Einsatz. Dieses Konzept ist aber sehr unflexibel, da, falls weitere auf anderen Technologien basierende Dienste in das System hinzugefügt werden, ihre nahtlose Einbindung unmöglich ist. Alle Vorteile dieses Ansatzes wurden zudem mit der Entwicklung eines eigenen Web-Browsers beibehalten.

Für die Speicherung der Benutzerdaten wurden im Rahmen dieser Arbeit eine Benutzerdatenbank implementiert und die Benutzerverwaltung realisiert. Es wurde ein Model der Zentralen Benutzerverwaltung gewählt, bei der alle authorisierten Logins/Accounts von einem Systemadministrator in die Benutzerdatenbank eingetragen werden. Den Benutzern werden dann ihre Zugangsdaten mitgeteilt (Login und Passwort). Ein Benutzer kann sich unter Verwendung dieser Daten in das System einloggen und wird anhand seines Logins, das nur einmal in der Datenbank vorkommt, verwaltet. Vom Anwender können die Angaben zu seiner Person, die auch in der Benutzerdatenbank abgespeichert werden, jederzeit verändert werden.

Das System wurde für eine hohe Anzahl von Benutzern konzipiert (100 und mehr). Deshalb wurde auch eine Möglichkeit vorgesehen, die Datenbank auf einem eigenen Rechner zu installieren. Sie wird mit dem zentralen Server übers Netzwerk verbunden.

3.2 Konzeption des Dienstes zur Wiedergabe von Vorlesungen

Mit dem Dienst zur Wiedergabe von Vorlesungen wird dem Benutzer eine Möglichkeit bereitgestellt, auf die Vorlesungsinhalte, die bereits vorbereitet wurden, zuzugreifen. Dieser Dienst stellt die einfachste Art der Shared Applications dar. Es können zwar mehrere Benutzer auf die abgespeicherten Daten zugreifen, geändert werden können sie aber nur vom Administrator des Systems.

Weiterhin müssen die Inhalte spezifiziert werden, die dieser Dienst zur Wiedergabe von Vorlesungen abspielen können muss.

Bei den durchgeführten Tests hat sich gezeigt, dass es nicht ausreicht, nur das Video des Vortragenden abzuspeichern und dann bei Bedarf zu übertragen: Die Auflösung des zur Abspeicherung verwendeten MPEG-Datenformates beträgt bei einer ¼-PAL-Qualität nur 352x288 Pixel. Aufgrund dieser niedrigen Auflösung und der hohen Kompressionsrate von ca. 50:1 können Details wie Texte, Diagramme usw. nicht lesbar abgebildet werden.

Es ist deshalb notwendig, nicht nur die Audio-/Videoinformation zu übertragen, sondern auch synchron dazu die grafische bzw. alphanumerische Information, die vom Vortragenden präsentiert wird. Für die Realisierung kann es dabei vorteilhaft sein, wenn die Inhalte von Vorlesungen bereits in Form von HTML-Dokumenten existieren, die den Interessenten im Netz zur Verfügung stehen. Für die Nutzung mit dem im Rahmen dieser Arbeit entwickelten System brauchen diese Dokumente nur minimal verändert (neu formatiert) werden.

Die Dokumenteninhalte müssen noch mit dem Videostrom synchronisiert werden:
Dazu wird der Audio-/Videostrom der Vorlesungsaufnahme in Fragmente zerlegt, die jeweils passend zum Inhalt dargestellter Dokumente abgespielt werden. Der Benutzer kann dann selbst zwischen diesen Vorlesungsabschnitten navigieren.

Die Zerlegung des Videostroms kann auf zwei Weisen realisiert werden:

1. Während der Aufnahme drückt der Vorlesende beim Umblättern auf eine Taste eines Zeit-messenden Programms, sodass damit die Werte der Zeitabstände zur Synchronisation abgespeichert werden.

2. Bei der Nachbearbeitung wird der Videostrom nachträglich in Videoabschnitte zerlegt, passend zu den dargestellten Dokumenteninhalten.

Da die Videoaufnahmen in jedem Fall nachbearbeitet werden, wurde die zweite Lösung gewählt. Ausserdem fällt dabei der Mehraufwand für den Entwurf eines speziellen Programms zur Synchronisation weg.

3.3 Chat-Dienst

Für die Kommunikation zwischen den Teilnehmern wird in das System ein Chat-Dienst integriert, der im Rahmen dieses Projektes entwickelt wird [RENZ00]. Er stellt eine textbasierte Konferenzkomponente dar, wie bereits im Abschnitt 2.1 beschrieben.

Die allgemeinen Anforderungen an einen komfortablen Chat-Dienst sind:

- mehrere Chatgruppen werden verwaltet
- ein Benutzer erstellt nach belieben entweder eine eigene Gruppe oder kann einer bestehenden Gruppe beitreten
- die von verschiedenen Nutzern eingegebenen Beiträge werden möglichst mit verschiedenen Farben dargestellt, um die Lesbarkeit und die Differenzierung zwischen den Beiträgen einzelner Benutzer zu erleichtern.

Ein Chat-Dienst besitzt eine Client/Server-Architektur. Das heisst, auf den Rechnern von Anwendern kommen die Chat-Service-Nutzer (Clients) zum Einsatz, mit denen Texte eingegeben und Beiträge anderer Benutzer dargestellt werden. Alle diese Chat-Service-Nutzer sind mit einem Chat-Service-Anbieter (Server) verbunden, an den sie die neu eingegebenen Daten übermitteln, und der dann die Verteilung von diesen Daten an andere Teilnehmer der Gesprächsrunde erledigt. Ausserdem beinhaltet der Chat-Service-Anbieter auch sein eigenes Sitzungsmanagement, das die Verwaltung der sich mit dem Chat-Service-Anbieter in Verbindung befindlichen Chat-Service-Nutzer und der Gesprächsgruppen übernimmt.

Bei einem Chat-Dienst wird reine alphanumerische Information übertragen. Hier folgt die Abschätzung des möglichen Nutzdatendurchsatzes ohne Overhead; Header der Pakete werden dabei nicht berücksichtigt (Tabelle 3.1):

Als ein Beispiel wird im Folgenden angenommen, dass ein Benutzer etwa 5 Zeichen pro Sekunde eingeben kann.

Bei einer Benutzerzahl von 100 Personen ergibt sich folgende Rechnung:

Wird Unicast zur Datenverteilung verwendet, werden die Daten vom Chat-Service-Anbieter auf alle 100 Benutzer verteilt. Somit ergibt sich eine Belastung von etwa 4 Kbit/s für einen Benutzer, oder, falls alle gleichzeitig schreiben, max. 400 Kbit/s

Die Ergebnisse der Testläufe haben gezeigt, dass, wenn sich mehr als 20 Benutzer in einer Chat-Gruppe befinden, die Kommunikation zwischen ihnen unübersichtlich wird. Deshalb hier eine veränderte Rechnung mit 100 Benutzern und 20 Benutzer/Gruppe:

Ein Benutzer verursacht in einer Gruppe einen Datenstrom von 5*20*8=0,8 Kbit/s
Bei 20 Benutzern pro Gruppe und 5 Gruppen ergibt sich ein Datenstrom von
Max. 80 Kbit/s

Angaben	**100 Benutzer, 1 Gruppe**	**100 Benutzer, 5 Gruppen**
Schreibegeschwindigkeit eines Benutzers	5 Zeichen/s	5 Zeichen/s
Anzahl der Benutzer in einer Gruppe	100	20
Belastung des Netzwerks durch einen Benutzer	4 Kbit/s	0,8 Kbit/s
Datendurchsatz pro Gruppe, max.	400 Kbit/s	16 Kbit/s
Gesamte Belastung des Netzwerks, max.	400 Kbit/s	80 Kbit/s

Tabelle 3.1: Datendurchsatz eines Chat-Service-Anbieters (Nutzdaten ohne Overhead)

Da der Datendurchsatz eines Chat-Service-Anbieters, wie oben aufgeführt, auch bei einer hohen Anzahl von Benutzern niedrig bleibt, kann er sich auf einem Rechner mit dem zentralen Server befinden. Dabei bleibt der Chat-Service-Anbieter vom zentralen Server unabhängig, und auch die Kommunikation zwischen den beiden Servern ist nicht vorgesehen, die Synchronisation erfolgt indirekt über die Chat-Service-Nutzer. Der Chat-Service-Anbieter muss lediglich zum Zeitpunkt des Starts des Systems ebenfalls gestartet werden und läuft dann parallel weiter.

Um die Kontrolle der Abläufe im Chat-Dienst zu gewährleisten, bauen die Chat-Service-Nutzer zusätzlich zur Steuer- und Datenverbindung zum Chat-Service-Anbieter auch eine weitere Verbindung zum zentralen Server auf. Das geschieht gleich nach dem Aufruf des Dienstes aus der zentralen Applikation, die auf der Benutzerseite läuft. Der zentrale OVID-Server trägt dann einen Vermerk in die Tabelle des Sitzungsmanagements ein, dass ein entsprechender Dienst gestartet wurde.

Wenn ein Anwender seine Chat-Sitzung beendet, wird eine Nachricht an den zentralen OVID-Server geschickt, die es ihm mitteilt. Der zentrale Server teilt dies dann seinerseits der zentralen Anwenderapplikation mit. Wenn der Anwender das Hauptprogramm verlässt, ohne vorher seine Chat-Sitzung ordnungsgemäss zu schliessen, wird beim Beenden des Hauptprogramms eine entsprechende Nachricht an den zentalen OVID-Server generiert, welches diese Nachricht dann an den Chat-Service-Nutzer weiterleitet und zum automatischen Abbruch der Chat-Sitzung führt. Wenn die Kommunikation zwischen den Teilnehmern einer Chat-Sitzung unerwünscht ist, kann der zentrale Server die bestehende Sitzung unterbinden.

Somit lässt sich, dank der zusätzlichen Verbindung vom Chat-Service-Nutzer zum zentralen Server, eine vollständige Synchronisationen zwischen den Teilkomponenten des Gesamtsystems erreichen. In Abb. 3.2 wird die Kommunikation zwischen den Teilkomponenten beim Benutzen eines Chat-Dienstes schematisch dargestellt.

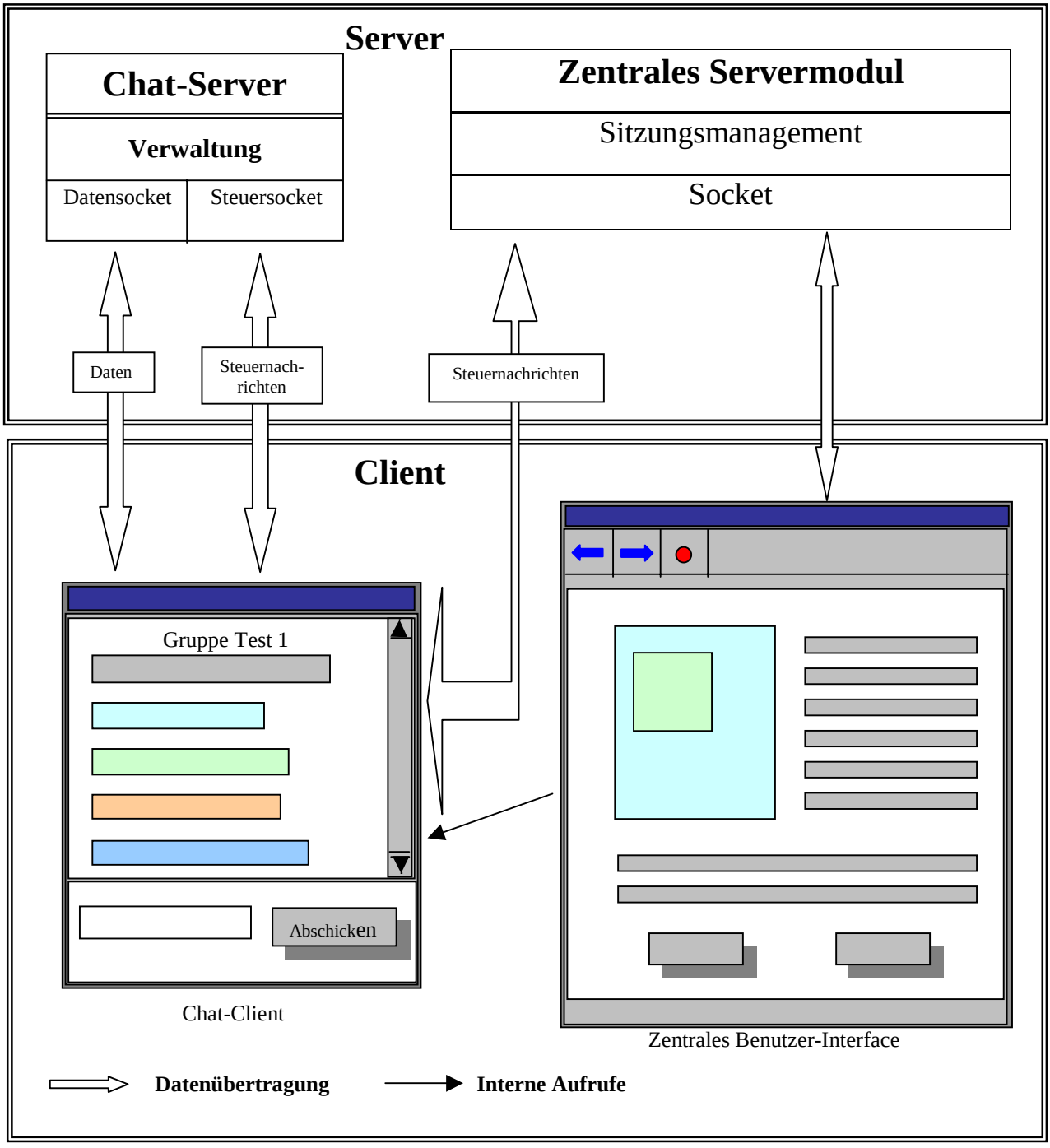

Abb. 3.2: Schematische Darstellung der Kommunikation zwischen Teilkomponenten beim Benutzen eines Chat-Dienstes

3.4 FTP und E-Mail

Für die komfortable Arbeit mit einem Distance-Learning-System sollen die Teilnehmer auch eine Möglichkeit haben, untereinander Dateien auszutauschen und E-Mails zu verschicken. Die dafür eingesetzten Komponenten benötigen keine zusätzliche Anbindung an den zentralen Server. Es soll lediglich eine Möglichkeit zum Aufruf diverser Dienste aus der zentralen Client-Application gegeben sein. Für die Zwecke der besseren und nahtlosen Integration wurde ein FTP-Client neu entwickelt [RENZ00]. Als E-Mail-Dienst kommt Microsoft Outlook Express oder ein anderer E-Mail-Client zum Einsatz. Der E-Mail-Dienst soll im allgemeinen dann verwendet werden, wenn der Teilnehmer nicht über den Dienst zum Verschicken der Kurzmitteilungen erreicht werden kann, d.h. wenn dieser Teilnehmer nicht mit dem System arbeitet. Er kann z.B. genutzt werden, um eine solche Person zur Zusammenarbeit einzuladen.

3.5 Videokommunikation

Eine bequeme Art der Kommunikation stellt die Audio-/Videokommunikation dar, denn sie kommt der gewohnten zwischenmenschlichen Kommunikation am nähesten. Für die Bereitstellung der Audio-/Videokommunikation in der OVID-Distance-Learning-Umgebung wird ein Online-Audio-/Video-Kommunikationssystem für PAL-Qualität verwendet [NGU98].

Die Zielsetzung bei der Entwicklung des Online-Audio-/Video-Kommunikationssystem war es zu untersuchen, ob und wie Übertragungen bei höchstmöglicher Video- und Audioqualität mit herkömmlichen PC-Systemen erreicht werden können. Zum Zeitpunkt des Entwicklungsbeginns waren das Intel-Pentium PC’s mit 120MHz Taktfrequenz. Dabei wurden Möglichkeiten gefunden, die Übertragung wechselseitig in PAL-Auflösung und CD-Audioqualität zu realisieren. Um diese hohe Qualität zu erreichen, wurden die Videodaten im MJPEG-Format komprimiert. Dafür wurden handelsübliche Videoschnittadapter eingesetzt, die auf einem Codec der Firma Zoran basieren. Bei diesen Adaptern beträgt die Umschaltzeit zwischen den Betriebsarten Komprimieren/Dekomprimieren mehrere Sekunden, was den Einsatz eines einzelnen Adapters sowohl für die Aufnahme als auch für die Wiedergabe im Vollduplexbetrieb verhindert. Deshalb müssen für den Vollduplexbetrieb zwei Adapter in

einem Rechner betrieben werden. Der Grafikadapter muss dabei einen sogenannten Overlay-Modus unterstützen, was aber bei allen modernen Grafikadaptern der Fall ist.

Für die hohe Qualität der Übertragung werden ein Breitbandnetz (ATM oder Fast-Ethernet) oder ein Ethernet-LAN, das der Videokommunikation exklusiv zur Verfügung steht, vorausgesetzt. Über eine Standard-Ethernet-Verbindung kann unter Umständen bis ¼ PAL-Auflösung übertragen werden. Das System setzt auf den Standard-IP-Stack auf.

Das Audio-/Video-Kommunikationssystem wird in der Distance-Learning-Umgebung für die Bereitstellung der Punkt-zu-Punkt Verbindungen zwischen zwei Benutzrn verwendet. Dabei kann der Initiator der Videokommunikation einen Request an die gewünschte Person schicken, und wenn der Request akzeptiert wird, wird automatisch eine Videoverbindung aufgebaut.

3.6 Whiteboard

Wenn mehrere in das System eingeloggte Benutzer an einem gemeinsamen Projekt arbeiten, kann dadurch die gemeinsame Bearbeitung eines Dokumentes notwendig werden. Die Programme, die mehreren Personen eine gemeinsame und gleichzeitige Bearbeitung eines grafischen Dokumentes erlauben, sind allgemein unter dem Begriff ‚Whiteboard' bekannt.

Ein Whiteboard-Dienst bietet die Funktionalität eines einfachen Bildbearbeitungsprogramms an, und die neuen Elemente werden zeitgleich mit ihrer Eingabe oder mit minimalen Verzögerungen zwischen allen Teilnehmern einer Sitzung verteilt. Mehrere Whiteboard-Dienste wurden auf ihre Eignung zum Einsatz mit dem Distance-Learning-System untersucht [RENZ00].

Einer der möglichen Kandidaten ist das Tool „mash-MB", das an der Berkley University entwickelt wurde. Es wird auch von den Entwicklern zum Einsatz unter Windows98 und NT empfohlen. Die Datenübertragung basiert bei diesem Tool auf dem UDP-Transportprotokoll. Für den Verbindungsaufbau werden beim Start der Sitzung eine Multicastadresse für diese Sitzung und der Port benötigt.

4 Implementierung

In diesem Kapitel wird die Umsetzung des Konzepts des Distance-Learning-Systems OVID erklärt.

4.1 Systemanforderungen

Als Entwicklungsplattform wurden wegen ihrer Verbreitung und Akzeptanzgrades bei den Anwendern Windows 95/98/NT/2000 ausgewählt.

- Für den Einsatz auf den Server-Rechnern wurde wegen ihrer hohen Stabilität und unter Berücksichtigung von Sicherheitsaspekten Windows NT/2000 gewählt
- Für den Endanwender kommen sowohl Windows 95/98 als auch Windows NT/2000 in Frage. Da die bereits entwickelte Online-Videokommunikationskomponente nur unter Windows 95/98 lauffähig ist, ist die vollständige Funktionalität z.Z. auch nur unter diesen Betriebsystemen gegeben.

Als Programmiersprachen standen Java und C++ zur Auswahl.

Die Performance von C++-Programmen ist generell und Prinzip-bedingt höher als bei Programmen, die mit Java geschrieben werden: Ein C++-Compiler liefert einen optimierten und auf die CPU zugeschnittenen Binärcode, die Java-Applets werden aber erst während der Laufzeit im Interpreter-Modus in den Maschinencode übersetzt und ausgeführt, um die Plattformunabhängigkeit zu erreichen.

Der Einsatz von Java bringt bei der Implememtierung dieses Distance-Learning-Systems keine Vorteile, denn das wichtigste Argument für Java – seine hohe Portabilität – kann hier nicht realisiert werden. Bei der Entwicklung wurde tief in das Systeminnere gegriffen und von den Windows-Eigenschaften Gebrauch gemacht (das Benutzerinterface wurde mit dem Browserkern des Microsoft Internet Explorers gekoppelt), so dass das fertige System nicht ohne weiteres auf ein anderes Betriebsystem portiert werden kann, was den Sinn des Einsatzes von Java in Frage stellt. Ausserdem werden in der Distance-Learning-Umgebung Dienste verwendet (z.B. Audio-/Videokommunikations-Dienst), die bereits für den Einsatz unter Windows entwickelt und optimiert wurden.

Als C++ - Compiler [STR92] wurde der Microsoft Visual C++ aus dem Microsoft Visual Studio 6.0 verwendet. Dieser Compiler von Microsoft bietet eine funktionelle Entwicklungsumgebung. Dank der Tatsache, dass er vom selben Hersteller wie das Betriebsystem kommt, ist er für Entwicklungsarbeiten unter Windows besonders geeignet und bietet Zugriff auf alle Systemressourcen.

Die Entwicklung erfolgte durchgehend objektorientiert. Es wurde bei der Entwicklung die MFC (Microsoft Foundation Classes) verwendet. Der Begriff MFC ist untrennbar mit Visual C++ verbunden. Dabei stellt die MFC-Bibliothek vorrangig eine Kapselung der Win32 API dar. Fast alle technischen Merkmale von Windows 98 bzw. NT sind in irgendeiner Form durch die MFC abgedeckt [YOU97].

Das Servermodul wird für den Einsatz an einer leistungsstarken Workstation (P II/III mit mind. 400 MHz, 128 MByte oder mehr Arbeitsspeicher) entwickelt, die über eine Netzanbindung mit hoher Bandbreite verfügt. Als Betriebsystem dient Windows NT. Dieses Modul verwaltet die Benutzer in einer Datenbank, vergibt Rechte zum Aufruf von zusätzlichen Diensten und führt das Sitzungsmanagement durch. Unter Sitzungsmanagement wird dabei die Kontrolle aller Aktivitäten des Benutzers ab dem Zeitpunkt der Anmeldung am System bis zum Verlassen des Systems verstanden. Auch wird dabei den zusätzlichen Diensten die angeforderte Information über die Benutzer des System für interne Verwaltungszwecke zur Verfügung gestellt.

Die Anforderungen an die Client-Hardware sind moderater als beim Server – ein handelsüblicher PC (die Tests wurden an einem PC mit Pentium 60 und 32 MB Speicher erfolgreich durchgeführt) ist dabei ausreichend. Wenn aber die Videokommunikationskomponente zum Einsatz kommen soll, werden mindestens eine Videoschnittkarte und ein leistungsfähiger Netzzugang benötigt. Die verwendete Videokommunikationskomponente ist zwar auch bei einer Bandbreite von ca. 10 Mbit/s lauffähig, welche auch in den herkömmlichen Ethernet-LANs erreicht wird, bei dieser hohen Belastung ist aber eine häufige Kollision von Paketen unvermeidbar und die Videokommunikation wird gestört. Deshalb soll, falls eine Videokommunikation über Ethernet erfolgt, das Netzwerk exklusiv der Videokommunikationskomponente zur Verfügung stehen, was aber aufgrund des Aufbaus des im Rahmen dieser Arbeit entwickelten Systems kaum möglich ist.

4.2 Client

Um die Integrität des Systems zu wahren wurde ein eigener Web-Browser (basierend auf dem Browserkern von Microsoft Internet Explorer) entwickelt, der die zusätzliche Funktionalität zur Kommunikation mit dem OVID-Servermodul und die Möglichkeit zum Starten von Zusatzdiensten besitzt. Dieser Browser dient dem Benutzer als eine Schaltzentrale und übernimmt die Funktionen eines zentralen Anwendermoduls. Der Web-Server kann sich dabei sowohl auf dem Rechner mit dem zentralen Servermodul als auch auf einem anderen Rechner befinden, je nachdem, wie hoch die Auslastung ist.

Die Vorlesungsinhalte werden im Fenster des für OVID entwickelten Web-Browsers dargestellt. Alle modernen Web-Browser, die unter Windows-Betriebsystemen laufen, besitzen auch die Funktion eines Containers für die ActiveX-Komponenten, die im Anhang E beleuchtet wurden. Der aktuelle Media Player von Microsoft, der für die Wiedergabe von den Multimedia-Daten (AVI- und MPEG-Videos, WAV-Audiodateien) zuständig ist, stellt nichts anderes, als ein ActiveX-Control dar. Das heisst, er kann sowohl als ein selbständig lauffähiges Programm ausgeführt werden (dabei wird ein neues Fenster für die Wiedergabe und für die Steuerelemente aufgemacht), als auch in einem Fenster eines Containers (in unserem Fall des Browsers) eingebettet werden (Abb. 4.1).

Abb. 4.1: Beispiel eines ActiveX-Dokuments in einem Browser-Fenster

Der Media Player 2 ist für die Wiedergabe von den Videodateien nicht nur wegen seiner ActiveX-Eigenschaften geeignet, sondern auch deshalb, weil er Streaming-fähig ist.
Unter Streaming versteht man die Möglichkeit, Daten während des Ladeprozesses gleichzeitig wiederzugeben. Das heisst, bereits dann, wenn der Ladevorgang noch nicht abgeschlossen ist, werden die Daten dekodiert und stückchenweise abgespielt. Wenn die Geschwindigkeit der Netzverbindung hoch genug ist, d.h. die Bandbreite höher ist, als die Grösse des Datenstroms, der abgespielt wird, kommt keine Stockung zu stande.

Im Rahmen dieser Arbeit wurde unter anderem auch ein GUI, das grafische Benutzerinterface, entworfen. Wie im Kapitel 3 bereits besprochen, wurde für den Einsatz am Client-Rechner ein Browser entwickelt, der auch die zusätzliche Funktionalität zur Kommunikation mit dem zentralen Server und für den Aufruf der weiteren Dienste bietet. Beim Entwurf des GUIs wurde an den Standard-Aufbau der MS-Fenster orientiert. Dazu wurde das Hauptfenster entsprechend gestaltet und die Bedienelemente wie Tasten und Menü wurden im oberen Bereich des Fensters plaziert. Die Tatsache, dass sich hinter diesem Programm die Funktionalität eines Web-Browsers verbirgt, ist dabei für den Endanwender nicht unbedingt ersichtlich, deshalb wurden praktisch alle von handelsüblichen Browsern bekannten Bedienelemente eingespart, lediglich die Tasten zum Vorwärts- und Rückwärtsnavigieren wurden implementiert.

Die Funktionsweise des Programms soll auch bei der Erstbenutzung ersichtlich sein, deshalb wurde als Startseite eine kurze Beschreibung integriert, wie das Programm zu bedienen ist.
Ein Fenster zum Zeitpunkt des Startes des Client-Moduls ist in Abbildung 4.2 dargestellt.
Für die Implementierung der Browser-Funktionen wurde die Klasse CHtmlView aus der MFC verwendet. Diese Klasse wurde erst in der Version 6 des Compilers realisiert. Sie beinhaltet praktisch die gesamte Funktionalität, die für die Entwicklung eines Web-Browsers notwendig ist. Für die auf dieser Klasse basierten Applikationen ist die Präsenz des Internet Explorers von Microsoft ab Version 4.0 im System erforderlich. Netscape und andere Browser werden z.Z. nicht unterstützt. Die Ergebnisse der Testläufe haben allerdings gezeigt, dass der Internet Explorer 4.0 einen schwerwiegenden Fehler in der Implementierung des so genannten Res-Protokols hat: Sobald die Information aus den Ressourcen-Files ausgelesen wird, gibt es bei ihm Darstellungsfehler, die die Nutzung des Internet Explorers 4.0 nur eingeschränkt möglich machen. Dieser Fehler macht den Einsatz des Res-Protokols

entsprechend seiner Spezifikation praktisch unmöglich, obwohl es zu keinen allgemeinen Schutzverletzungen kommt. Bei Microsoft ist dieser Fehler auch bekannt, aber er wurde erst in der Version 5.0 des Internet Explorers behoben.

Abb. 4.2: Wiedergabe des grafischen Eröffnungsfensters des zentralen Benutzermoduls

Zur Realisierung der Kommunikation werden Sockets in Verbindung mit dem Client/Server-Modell im Distance-Learning-System OVID verwendet. Dabei hat die MFC eine eigene Klasse für die Socket-Programmierung [QUI95], die CSocket-Klasse. Bei der Implementierung der Kommunikation zwischen Server und Client wurde eine Besonderheit der MFC-Klassen benutzt, und zwar lassen sich alle Instanzen von den MFC-Klassen serialisieren. Unter Serialisierung versteht man hier die Möglichkeit, ein Objekt dauerhaft in einem File abzuspeichern und anschliessend wieder neu zu laden. Diese Eigenschaft kann sowohl zum

Abspeichern von Objekten auf der Festplatte als auch für ihre Übertragung über das Netzwerk verwendet werden. Denn eine Socket-zu-Socket-Verbindung kann auch als ein File betrachtet werden, das wahlweise für Lesen oder Schreiben geöffnet wird. Dadurch wird die Definition eines speziellen Kommunikationsprotokols überflüssig: Ein Objekt wird auf einer Seite der Verbindung abgespeichert und auf der anderen Seite wieder als ganzes ausgelesen. Die einzige Bedingung ist dabei die Notwendigkeit, diese Klassen sowohl auf dem Server als auch auf dem Client zu definieren und zu beschreiben. Dieser Mechanismus funktioniert sogar

Programmiersprachenübergreifend, d.h. wenn ein C++ - Objekt abgespeichert wird, kann es an der anderen Seite als Java-Objekt ausgelesen werden, wenn die Implementierung gleich bleibt, was aber nur mit den Programmiersprachen aus dem Microsoft Visual Studio, d.h. mit Visual C++ und Visual Java funktioniert.

4.3 Kommunikationsabläufe

Bei der Installation des zentralen Client-Moduls muss zuerst die Adresse des Server-Rechners eingetragen werden, auf dem das zentrale Servermodul gestartet wird. Beim Starten einer Sitzung wird nach der Eingabe des Logins und des Passworts eine Verbindung zum Port 700 dieses Servers aufgebaut. Dann wird vom Client an den Server ein Objekt übertragen, das eine Instanz der Klasse CConReq darstellt. Dieses Objekt beinhaltet die Login-Information, die der Benutzer beim Verbindungsaufbauversuch eingegeben hat (Abbildung 4.3). Kommt es bei dem Server an, wird dieses Objekt zuerst gebeten, sich zu identifizieren. Der Server erwartet zu keinem Zeitpunkt Objekte eines bestimmten Typs. Die Überprüfung, was für ein Objekt gerade empfangen wurde, führt nicht der Server durch, sondern ein Objekt identifiziert sich selbständig und wird dann dementsprechend behandelt. Deshalb ist allen Klassen, deren Objekte zwischen dem Client und dem Server ausgetauscht werden, gemeinsam, dass sie alle Kinder einer Vaterklasse sind und eine virtuelle Funktion zu ihrer Identifizierung beinhalten.

```
void CMainFrame::On_B_Verbinden()
{
 if (m_pArchiveOut == NULL) // Es besteht noch keine Verbindung
 {
      char str_log[16],str_pas[16];
      CString sp;
      .
      .  Auslesen des Logins und des Passworts
      .
      if (m_Login==""||sp=="") // Felder Login oder Passwort leer
      {
       AfxMessageBox("Die Eingabe ist nicht korrekt !");
      }
      else{
            if (m_pArchiveOut == NULL) // Es besteht noch keine Verbindung
            {
              ConnectSocket(); // Aufruf der Funktion zum Verbindungsaufbau
              if (m_pArchiveOut != NULL) // Verbindung steht
                  {
                    CMyObj* pReq = new CConReq(m_Login,sp);
                        //Aufbau eines CConReq-Objekts
                    *(m_pArchiveOut) << pReq; // Request wird verschickt
                    m_pArchiveOut->Flush();
                    CMyObj *pUrl=NULL;
                    *(m_pArchiveIn) >> pUrl; // Ein URL wird empfangen
                    if (pUrl->GetUrl() != "") // Anmeldung erfolgreich
                     {
                   ((CTest3View*)GetActiveView())->Navigate(pUrl->GetUrl());
                        //Browser wird auf den neuen URL umgeschaltet
                   ((CTest3View*)GetActiveView())->PermGoBack(1);
                        // Button zum Navigieren wird freigeschaltet
                   CEdit* lo = (CEdit*)m_wndDlgBar.GetDlgItem(IDC_LOG);
                   lo->SetWindowText("");
                   CEdit* pa = (CEdit*)m_wndDlgBar.GetDlgItem(IDC_PAS);
                   pa->SetWindowText("");
                        // Felder Login und Passwort werden gelöscht
                     }
                   else // Anmeldung fehlgeschlagen, Passwort falsch
                        {
                        AfxMessageBox("Login falsch");
                        SockShutDown(); // Verbindung wird gekappt
                        // und das Programm in den Anfangszustand versetzt
                      }}}
      else{
            AfxMessageBox("Die Verbindung besteht bereits");
      }}}
 else // Der Benutzer will seine Sitzung beenden und die Verbindung trennen
{
      CMessage* pMes;
      pMes = new CMessage("Shutdown"); // Eine Shutdown-Nachricht wird vorbereitet
      *(m_pArchiveOut) << pMes; // Die Nachricht wird an den Server verschickt
      m_pArchiveOut->Flush();
      SockShutDown(); // Verbindung wird gekappt
      ((CTest3View*)GetActiveView())->LoadFromResource(IDR_HTML1);
            // Laden des Begrüssungfensters
      ((CTest3View*)GetActiveView())->PermGoBack(0);
      ((CTest3View*)GetActiveView())->PermGoForward(0);
            // Buttons zur Navigation werden abgeschaltet
}}
```

Abb. 4.3: Quelltext für die Erzeugung des Verbindungsaufbaus zum OVID-Server

Während des Startes baut der Server eine Verbindung zu der Datenbank auf, in der die Benutzerdaten abgespeichert sind. Zur Implementierung der Kommunikation mit der Datenbank standen zwei Alternativen zur Auswahl:

1. DAO (Data Access Objects) stellt eine Objekthierarchie dar, mit deren Hilfe die Microsoft Jet-Engine gesteuert werden kann. Die Jet-Engine ist die Datenbankmaschine von Microsoft Access.

2. ODBC-Standard oder Open Database Connectivity definiert ein Protokoll zwischen Betriebsystem und relationalen bzw. nicht-relationalen Datenbank-Management-Systemen. Die Anwendung wird dabei von der Art der Realisierung und somit vom Hersteller der Datenbank entkoppelt.

Obwohl letztendlich die Access-Datenbank von Microsoft zum Einsatz kam, wurde bei der Implementierung die ODBC-Schnittstelle gewählt, da sie eine viel höhere Flexibilität aufweist und den Umstieg auf eine andere Datenbank ermöglicht.

Nach dem Empfangen einer Instanz der Klasse CConReq werden die darin enthaltenen Daten mit denen in der Datenbank verglichen. Falls die Daten übereinstimmen, wird der Anmeldevorgang abgeschlossen und dem Client wird eine Instanz der Klasse CObjUrl zugeschickt, die die Information über eine URL trägt, mit der der Client eine Verbindung aufbauen soll (Abbildung 4.3). Dann wird die Information über den Client in die Tabelle der bestehenden Verbindungen aufgenommen, die vom Sitzungsmanagement des Servers geführt wird. Wählt sich ein Benutzer zum ersten mal in das System ein, wird er ausserdem gebeten, Angaben zu seiner Person anzugeben. Ein entsprechendes Dialogfenster zeigt Abb. 4.4

Abb. 4.4: Dialogfenster für die Eingabe der Angaben zur Person

Falls die Login-Daten inkorrekt waren, wird dem Client eine leere Instanz der Klasse CObjUrl zugeschickt, was vom Client als ‚Verbindungsaufbau fehlgeschlagen' interpretiert wird. Dann wird die Verbindung unterbrochen und das entsprechende Socket des Servers freigegeben (Abb. 4.3).

Wenn auf dem Client ein Objekt der CObjUrl-Klasse ankommt, wird die Information über die URL, die darin enthalten ist, ausgewertet. Dann wird diese URL dem internen Web-Browser mitgeteilt, und es wird eine Verbindung zur Einführungsseite der Distance-Learning-Veranstaltungen hergestellt und der Inhalt dieser Seite im Hauptfenster dargestellt. Ab diesem Zeitpunkt kann der Benutzer zwischen verschiedenen Veranstaltungen navigieren, die auf dem Web-Server abgespeichert und miteinander mit Hilfe von Hyperlinks verknüpft sind.

Gleich nach dem Objekt der CObjUrl-Klasse wird vom Server ein Objekt der Klasse CButMng erstellt und an den Client geschickt. Auf dem Client werden durch das Empfangen und Auswerten dieses Objekts Tasten für den Aufruf der Zusatzdienste freigeschaltet, die zum Zeitpunkt des Programmstarts abgeschaltet sind (Abbildung 4.5). Dabei beinhaltet dieses Objekt eine 6-Bit Maske, deren Bits die entsprechenden Tasten (Dienste) ein- oder abschalten. Es wird vom Server festgelegt, welche Dienste zu welchem Zeitpunkt freigegeben werden dürfen. Die Nachricht zum Ein- bzw. Abschalten kann zu einem beliebigen Zeitpunkt und beliebig oft geschickt werden.

```
if (pObj->Ident() == SwitchButton) //Angekommenes Objekt identifiziert sich
      {
            int but;
            but=pObj->GetData(); // Maske wird ausgelesen
            m_Chat=but & 1;
            m_Ftp=(but & 2)/2;
            m_Mail=(but & 4)/4;
            m_Nachricht=(but & 8)/8;
            m_Video=(but & 16)/16;
            m_Wb=(but & 32)/32;
      }
                        .
                        .
                        .
void CMainFrame::OnUpdateChat(CCmdUI* pCmdUI) // Wird vom System aufgerufen
{
  pCmdUI->Enable(m_Chat != 0?1:0); //Button wird ein- bzw. abgeschaltet
}
// Ähnlich Funktionen werden auch zum steuern von anderen Tasten aufgerufen
```

Abb. 4.5: Quelltext zur Freigabe von zusätzlichen Diensten

Wenn eine Taste zum Starten eines zusätzlichen Dienstes freigegeben wurde, wird beim Anklicken dieser Taste durch den Aufruf der WinExec-Funktion ein neuer Prozess mit dem entsprechenden Dienst gestartet.

Wird am Ende der Sitzung vom Benutzer die Taste ‚Trennen' angeklickt, wird an den zentralen Servermodul eine Instanz der Klasse CMessage mit dem Inhalt ‚Shutdown' geschickt. Die Klasse CMessage kann eine einfache Nachricht beinhalten und wird für den Austausch von Steuernachrichten zwischen Client und Server verwendet. Nach Empfang der Nachricht ‚Shutdown' wird die Information über die bestehende Verbindung aus der Tabelle der Verbindungen herausgenommen und die Verbindung wird beendet. Ausserdem werden, falls erforderlich, auch an zusätzliche Dienstprogramme, die zum selben Anwender gehören und nicht vorher beendet wurden, die entsprechenden ‚Shutdown'-Nachrichten geschickt. Das Hauptprogramm des Benutzers wird dabei in den Anfangszustand gesetzt.
Auch wenn die Verbindung nicht ordnungsgemäss getrennt, sondern das Ausführen vom Programm her einfach unterbrochen wird, wird das Absenden der ‚Shutdown'-Nachricht, die zur korrekten Verbindungstrennung führt, durch den Aufruf eines entsprechenden Destruktors garantiert.

4.4 Implementierung der Dienste

Die Zusatzdienste wurden so implementiert, dass, wenn ein Dienst eine Client/Server-Architektur besitzt, wie z.B. der Chat-Dienst, der entsprechende Server (hier Chat-Server) unabhängig vom zentralen Servermodul läuft. Dabei verschickt der zentrale Server Steuernachrichten an die Zusatzdienste. Diese bauen dafür, falls erforderlich, einen Kommunikationskanal mit dem zentralen Servermodul.

Bis auf einen, den Nachrichtenaustausch-Dienst, sind alle anderen Dienste selbständige Programme. Dabei garantiert die Anwendung von Visual C++ bei den Diensten, die neu entwickelt wurden [RENZ00], einen nahtlosen Übergang zwischen dem Hauptprogramm und den zusätzlichen Modulen. Der Benutzer kann beim Aufruf eines Dienstes nicht unterscheiden, ob es sich dabei um ein eigenständiges Programm oder um ein neu geöffnetes Fenster des Hauptprogramms handelt. Dementsprechend müssen diese Dienste (Chat, WhiteBoard usw.) mit dem Hauptprogramm synchronisiert werden.

Bei den FTP- und Mail- Diensten gestaltet sich die Synchronisation zwischen ihnen und dem Hauptmodul einfach: Es wird beim Starten des Dienstes ein neuer Prozess kreiert, über den das Hauptmodul die Kontrolle behält. D.h. diese Prozesse können, falls erforderlich, vom Hauptprozess abgeschaltet werden, falls die Sitzung beendet wurde und die Verbindung zwischen dem Hauptmodul des Clients und dem zentralen Servermodul getrennt wurde. Eine Weitergehende Kontrolle über FTP- und Mail-Dienste wie z.B. eine Möglichkeit der Kommunikation zwischen ihnen und dem Hauptmodul ist nicht gegeben und auch nicht erforderlich.

Für den E-Mail-Austausch werden zwei Arten von Servern benötigt, die nicht unbedingt im System selbst installiert sein müssen. Der eine ist ein sogenannter POP3-Server, der es den E-Mail-Clients ermöglicht, die E-Mails abzuholen. Abgeschickt werden die E-Mails über einen SMTP-Server. In dem Distance-Learning-System können auch die Server, die für den E-Mail-Austausch bereits eingerichtet sind, weiterverwendet werden.

Für den FTP-Dienst ist es sinnvoll, einen entsprechenden FTP-Server auf einem der für das Distance-Learning-System verwendeten Server-Rechnern zu installieren. Dieser Server wird dann vom Administrator des Systems verwaltet, und die Inhalte werden von dem Administrator regelmässig aktualisiert und/oder überprüft.

Beim Starten des Dienstes ‚Nachrichtenaustausch' wird an den Server zuerst eine Anfrage in Form einer Instanz der Klasse CMessage, also eine kurze Steuernachricht mit dem Inhalt ‚GetUserList' geschickt. Der Server akzeptiert diese Nachricht als eine Aufforderung, dem Client eine komplette Liste aller im Moment angemeldeten Benutzer des Systems zuzuschicken. Diese Liste wird anhand der vom Sitzungmanagement geführten Tabellen erstellt und mit ihr wird ein Objekt der Klasse CStringList instanziert. Wenn dieses Objekt vom Client empfangen wird, wird dem Benutzer eine Liste aller möglichen Kommunikationspartner präsentiert (Abb. 4.6). Er wählt dann einen Partner aus dieser Liste aus und gibt die Nachricht ein, die anschliessend verschickt werden soll. Mit dieser Nachricht und dem UserID des Empfängers wird dann ein Objekt der Klasse CInfoMsg gefüllt. Wenn der Server ein solches Objekt zugeschickt bekommt, ersetzt er im Inneren des Objektes die Adresse des Empfängers durch die Adresse des Absenders und verschickt dieses Objekt weiter an den endgültigen Empfänger, dem eine entsprechende MessageBox präsentiert wird.

```
void CMainFrame::On_B_Nachricht() // Der Nutzer möchte eine Nachricht verschicken
{
      CMessage* pMes;
      pMes = new CMessage("GetUserList"); // Nachricht GetUserList wird erzeugt
      *(m_pArchiveOut) << pMes; // und anschliessend verschickt
      m_pArchiveOut->Flush();
      *(m_pArchiveIn) >> m_pUserList; // die Liste aller Benutzer wird empfangen
      CMsgDlg vDlg;
      vDlg.DoModal(m_pUserList,m_pArchiveOut); // und das Dialogfenster geöffnet
}
                  .
                  .
                  .
void CMsgDlg::OnOK() // Der Nutzer clickt OK-Taste an
{
      UpdateData(TRUE);
      CComboBox *pBox = (CComboBox*)GetDlgItem(IDC_COMBO_USERLIST);
      CString SlctUser;
      pBox->GetWindowText(SlctUser); // Name des selektierten Benutzers wird ausgelesen
      CMyObj* pMsg=new CInfoMsg(SlctUser,m_Nachricht);
        // Objekt der Klasse CInfoMsg wird mit dem Namen der Gegenstelle und der Nachricht instanziert
      *(m_pArchiveOut) << pMsg; // Das Objekt wird verschickt
      m_pArchiveOut->Flush();
      CDialog::OnOK();
}
```

```
if (pObj->Ident() == Message) // Nachricht kommt bei dem Empfänger an
      {
            CString name,str;
            pObj->GetData(name,str); // Daten werden augelesen
            AfxMessageBox("Nachricht von "+name+": "+str);
            // Eine entsprechende Meldung wird dem Benutzer präsentiert
      }
```

Abb. 4.6: Quelltextauszug zum Erstellen und Versenden einer Nachricht

Abb. 4.7 zeigt den Bildausschnitt beim Absender zum Zeitpunkt der Eingabe der Nachricht.

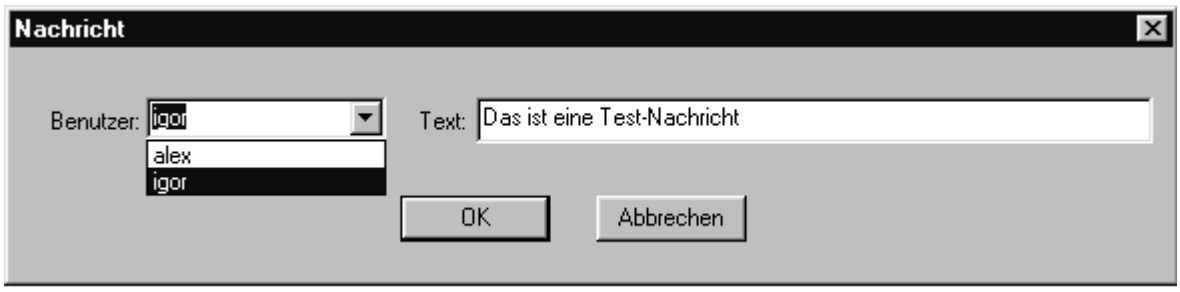

Abb. 4.7: Dialogfenster bei der Eingabe der Nachricht

In Abbildung 4.8 wird die Nachricht dargestellt, die dem Empfänger präsentiert wird.

Abb. 4.8: Dialogfenster zur Benachrichtigung des Empfängers

Wie bereits im Kapital 3 erwähnt, wird für die Videokommunikation zwischen den Benutzern des Systems ein Online-Audio-/Video-Kommunikationssystem verwendet [NGU98]. Für den Einsatz in der Disatnce-Learning-Umgebung wurde das vorliegende Programm modifiziert. Als Eingabeparameter benutzt das Programm die IP-Adresse der Gegenstelle, mit der eine Audio-/Videoverbindung hergestellt werden soll. Diese Eingabe erfolgt im Dialog-Modus. Deshalb war es notwendig, das vorhandene Programm zur Online-Audio-/Video-Kommunikation so umzuändern, dass es die Eingabe der IP-Adresse aus der Kommandozeile auslesen und auswerten kann. Des weiteren übernimmt das Hauptmodul die Verwaltung von den IP-Adressen. Dem Benutzer wird lediglich eine Liste mit den möglichen Kommunikationspartnern präsentiert, aus der er einen aussuchen kann. Anschliessend wird die Audio-/Videoverbindung automatisch aufgebaut. Der Prozess der Synchronisation zwischen Initiator der Audio-/Videokommunikation und Gegenstelle verläuft ähnlich wie beim Nachrichtenaustausch-Dienst: Es wird ein Objekt der Klasse CVideoRq erstellt, und über den Server an den ausgewählten Benutzer geschickt. Dabei fügt der Server in dieses Objekt noch die IP-Adresse des Initiators ein. Falls die Gegenstelle den Request akzeptiert, wird auch von ihr ein CVideoRq-Objekt an den Initiator verschickt, wobei der Server in diesem Fall die IP-Adresse der Gegenstelle einfügt. Somit verfügen am Ende der Synchronisationsphase sowohl der Initiator als auch die Gegenstelle wechselseitig über die jeweiligen IP-Adressen. Anschliessend wird die Verbindung gestartet. Wird der Request von der Gegenstelle abgelehnt, so wird eine entsprechende Nachricht an den Initiator geschickt.

Die Abbildung 4.9 zeigt das Dialogfenster mit der Liste der Sitzungsteilnehmer, das auf dem Rechner des Initiators geöffnet wird.

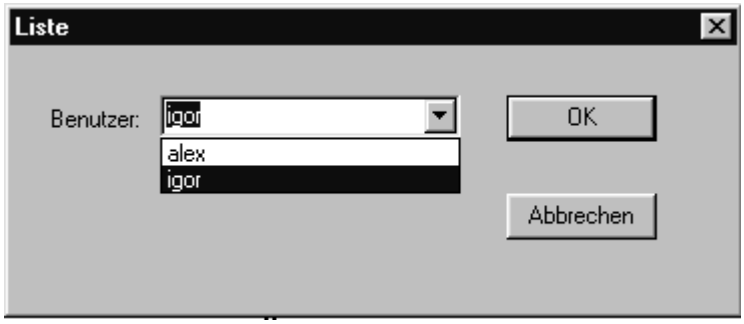

Abb. 4.9: Dialogfenster zur Auswahl des Kommunikationspartners

In Abb.4.10 ist die Anfrage zu sehen, die auf dem Rechner des Benutzers erscheint, der zur Teilnahme an einer Videokommunikationssitzung eingeladen wird.

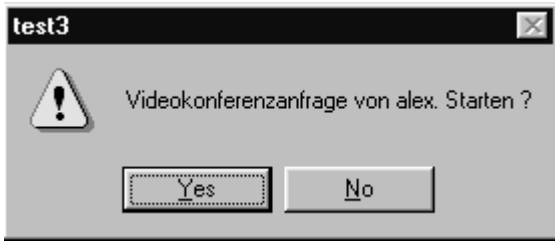

Abb. 4.10: Dialogfenster für die Einladung zur Teilnahme an der Videokommunikation

Für die Implementierung des Chat-Dienstes wurde eine Client/Server-Applikation entwickelt [RENZ00]. Es ist vorgesehen, dass der Chat-Server auf demselben Rechner wie das zentrale Servermodul läuft, was aber keine notwendige Bedingung darstellt. Dabei werden vom Server die Ports 701 und 702 belegt. Port 701 dient dem Austausch von Steuernachrichten mit den Chat-Clients. Über den Port 702 werden die Textzeilen, die von den Benutzern eingegeben werden, zwischen allen Benutzern verteilt.

Beim Start des Chat-Dienstes werden das Programm CHAT.EXE gestartet und die Adresse des Servers, gefolgt von der Nummer des Steuer- und des Datenports des Chat-Servers, in der Kommandozeile an dieses Programm übergeben (Abb. 4.11). Wird das Chat-Fenster vom Benutzer geschlossen, so wird ein Chat-Client-Destruktor ausgeführt, welcher eine bereits beschriebene ‚Shutdown'-Steuernachricht sowohl an das zentrale Servermodul als auch an den Chat-Server schickt.

```
void CMainFrame::On_B_Chat() //  Chat wird aufgerufen
{
STARTUPINFO StartupInfo;
memset(&StartupInfo,0,sizeof(STARTUPINFO));
StartupInfo.cb=sizeof(STARTUPINFO);
CString tmps="start "+m_Login+" "+m_ServerIP+" "+Port1+" "+Port2;
  //  Die Zeile mit den Parametern ( Name des Benutzers, IP des Servers, Port für den Steuer- und
  //  Datenkanal) wird aufgebaut
BOOL tmp=::CreateProcess //Der neue Prozess wird gestartet
 ("CHATCLIENT.EXE",
  tmps,
  NULL,
  NULL,
  FALSE,
  0,
  NULL,
  NULL,
  &StartupInfo,
  &ProcessInfo); //  Ein neuer Prozess wird kreiert
}
```

Abb. 4.11: Quelltextauszug zum Starten des Chat-Dienstes

5 Vergleiche mit JaTeK

Bei JaTeK [IRM99] handelt es sich um ein Teleteaching-Tool, das im Rahmen einer Kooperation zwischen der Technischen Universität Freiberg (Prof. K.Irmscher) und der Technischen Universität Dresden (Prof. A.Schill) entwickelt wurde. Die Arbeit am JaTeK-Projekt wurde 1997 gestartet. Das System befindet sich in einem fortgeschrittenen Stadium der Entwicklung [JaT99].

5.1 Beschreibung der Funktionalität zu JaTeK

Ziele des Projektes:

- Entwicklung und Evaluierung eines modularen Softwarepaketes zur Vor- und Nachbereitung von Studienmaterial für Vorlesungen, Seminare, Übungen und Praktika mit Hilfe von Internet-Komponenten
- Breiter Einsatz in unterschiedlichen Schichten der Aus- und Weiterbildung

JaTeK:Hauptmodul

Das JaTeK-System beinhaltet zwei Funktionen. Einerseits stellt es den Lehrenden Werkzeuge zur Erstellung interaktiv zu bearbeitender Übungsblätter sowie Schablonen (Templates) für die Integration von Lehrtexten in HTML-Format und Java-Applets für den Übungsbetrieb, zur Verfügung. Andererseits ermöglicht es den nutzerspezifischen Zugriff auf diese Lehrmaterialien über einen WWW-Server (Abbildung 5.1)

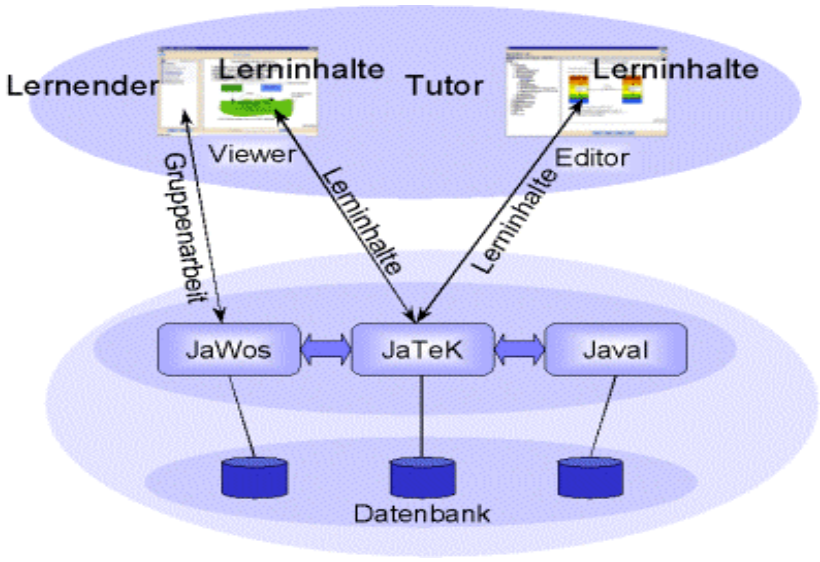

Abb. 5.1: Schematische Darstellung des JaTeK-Systems [JaT99]

JaWoS: Java Based Workgroup Support

Es handelt sich dabei um ein System zur Unterstützung von Übungsgruppen. Aufbauend auf die Broadcast- und Feedback-Möglichkeiten von JaTeK gibt JaWoS Studierenden die Möglichkeit, virtuelle Räume für Arbeitsgruppentreffen zu definieren und zu nutzen.
Das Modul eignet sich zur asynchronen sowie zur synchronen Interaktion.

Javal: Java Based Evaluation System

Javal ist ein Dienst zur Evaluierung von Lehrveranstaltungen. Es bietet die Möglichkeit, Evaluierungskriterien hinsichtlich des zur Verfügung gestellten Lehrmaterials zu definieren und empirisch zu überprüfen.

Sytem:

Das System JaTeK (Java Based Teleteaching Kit) besteht aus einem Server und einem Viewer mit integriertem Editor (Abb. 5.2)

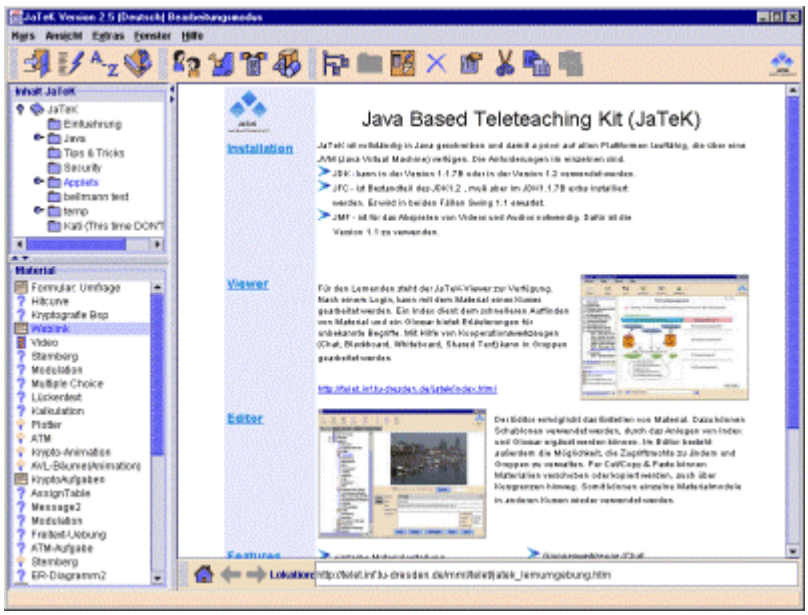

Abbildung 5.2: JaTeK-Benutzerinterface [JaT99]

JaTeK ist vollständig in Java geschrieben und damit auf allen Plattformen lauffähig, die über eine JVM (Java Virtual Machine) verfügen. Die Anforderungen im einzelnen sind:

- JDK - wird in der Version 1.3RC1 benötigt
- JMF - ist für das Abspielen von Videos und Audios notwendig. Dafür ist die Version 2.0 zu verwenden.

Viewer:

Für den Lernenden steht der JaTeK-Viewer zur Verfügung. Nach einem Login kann mit dem Material eines Kurses gearbeitet werden. Ein Index dient dem schnelleren Auffinden von Material und ein Glossar bietet Erläuterungen für unbekannte Begriffe. Mit Hilfe von Kooperationswerkzeugen (Chat, Blackboard, Whiteboard, Shared Text) kann in Gruppen gearbeitet werden (Abb. : 5.2).

Templates:

Der integrierte Editor ermöglicht das Erstellen von Material. Dazu können Schablonen verwendet werden, die durch das Anlegen von Index und Glossar ergänzt werden können (Abb. 5.3). Im Editor besteht außerdem die Möglichkeit, die Zugriffsrechte zu ändern und Gruppen zu ver- walten. Per Cut/ Copy&Paste können Materialien verschoben oder kopiert werden, auch über Kursgrenzen hinweg. Einzelne Materialmodule können in anderen Kursen verwendet werden.

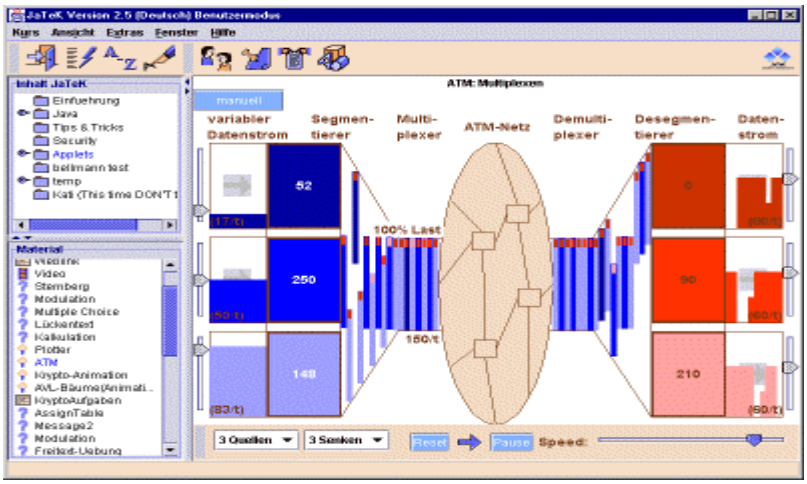

Abbildung 5.3: Templates in JaTeK [JaT99]

5.2 Konzeptvergleich

Im folgenden Abschnitt wird ein Vergleich mit dem im Rahmen dieser Arbeit konzipierten und implementierten Distance-Learning-System OVID durchgeführt.

Die Entwicklung von JaTeK begann im Jahre 1997. Zu diesem Zeitpunkt bot Java die besten Voraussetzungen, um eine multimediale und verteilte Online-Lernplattform zu entwickeln, die ausserdem systemunabhängig war. Da beim Entwurf von JaTeK diese Systemunabhängigkeit eine grosse Rolle spielte, mussten Abstriche in der Flexibilität gemacht werden. So können Im JaTeK-Viewer nur die HTML-Dokumente angezeigt werden, die für das System angepasst wurden. Inhalte, die auf neueren Technologien wie z.B. ActiveX basieren, können nicht dargestellt werden. Auch bei der Darstellung der Audio-/Videoinhalte ist die Flexibilität des JaTeK begrenzt: für die Wiedergabe wird eine spezielle Bibliothek JMF benötigt, die nur die wichtigsten Standards wie WAV oder AVI unterstützt. Mit dem im Rahmen dieser Arbeit entwickelten System ist das Abspielen aller Datentypen möglich, für die es die entsprechenden Plug-Ins für den Microsoft Internet Explorer gibt. Es können auch andere Kontexte wie z.B. PowerPoint-Folien, Java-Skripte, Java-Applets usw. bei der Erstellung der Lehrinhalte verwendet werden.

Seit 1997 haben sich neue Technologien entwickelt und etabliert. So wurden von Microsoft COM/DCOM vorgestellt. Ausserdem hat Microsoft die Vorherrschaft auf dem Web-Browser-Markt übernommen. Da bei dem Entwurf des Distance-Learning-Systems keine Portabilitätsaspekte mehr berücksichtigt wurden, konnte so eine bessere Integration mit dem Betribsystem als bei JaTeK erreicht werden. Mit dem Upgrade des Web-Browsers erfolgt beim OVID gleichzeitig ein Upgrade des Browserkerns, auf dem die Darstellung von HTML-Dokumenten basiert, und neue Technologien können somit gleichzeitig für die Erstellung der Dokumente mit den Vorlesungsinhalten verwendet werden.

Ein weiterer wichtiger Aspekt bei der Konzipierung von JaTeK war die Bereitstellung der Möglichkeiten zur Vorbereitung der Lehrinhalte. Zu diesem Zweck wird bei JaTeK ein spezieller Editor mit verschiedenen Templates verwendet. Bei dem in dieser Arbeit etwickelten Distance-Learning-System werden die Vorlesungsinhalte in Form von HTML-Dokumenten abgelegt und geladen. Da aber bereits komfortable HTML-Editoren existieren, können Lehrinhalte auch mit

diesen vorbereitet werden. Ausserdem steht der Entwicklung eines Editors der Lehrinhalte für das Distance-Learning-System nichts im Weg.

Die Verwaltung von verschiedenen Lehrinhalten wird bei JaTeK mit Hilfe einer Datenbank realisiert, die die URLs der entsprechenden HTML-Dokumente beinhaltet. Im vorliegenden Distance-Learning-System werden alle verfügbaren Veranstaltungen auf der Hauptseite in Form von Weblinks angeboten, sie werden auch für das Navigieren zwischen verschiedenen Abschnitten einer Veranstaltung benutzt, eine Datenbank erübrigt sich somit.

Der wichtigste Unterschied zwischen JaTeK und dem Distance-Learning-Tool OVID liegt in ihren unterschiedlichen Einsatzgebieten und folglich in verschiedener Implementierung:
JaTeK wurde für schmalbändige Netzwerke konzipiert und implementiert und beinhaltet deshalb keine Komponenten, die die Gruppenarbeit durch die sinnvolle Verwendung von hohen Bandbreiten vereinfachen oder komfortabler machen. Ausserdem dient JaTeK nur der Bereitstellung und der Wiedergabe von Lehrinhalten, es ist keine Unterstützung für virtuelle Veranstaltungen vorgesehen.

Das Distance-Learning-System OVID wurde dagegen für den Einsatz in Netzwerken mit hoher Bandbreite konzipiert, implementiert und optimiert und besitzt bereits in diesem Entwicklungsstadium eine Audio-/Videokommunikationskomponente. Sie erlaubt eine qualitativ hochwertige Kommunikation zwischen den Benutzern des Systems und kann später, um ein entsprechendes Sitzungsmanagement erweitert, für die Live-Übertragungen der Vorlesungen verwendet werden.

Die Konzepte von Distance-Learning-Systems und des JaTeK haben auch vieles gemeinsam:

- Client/Server-Architektur
- Lehrinhalte in Form von HTML-Dokumenten
- Ein browser-artiges Benutzerinterface auf der Client-Seite
- Gruppenverwaltung
- Erweiterbarkeit
- Gruppenwerkzeuge (Chat, Whiteboard, usw.)

6 Ergebnisse

Das Distance-Learning-System OVID wurde sowohl mit breitbandigen als auch schmallbandigen Netzwerken getestet.

Verwendete Hard- und Software:

Server-Rechner S1 (Zentraler Server)

- *CPU*: Pentium III 600 MHz
- *RAM*: 64 MB
- *Festplatte*: 20 GB
- *Betriebsystem*: Windows 98

Server-Rechner S2 (Zentraler Server)

- *CPU* : AMD K6-III 400 MHz
- *RAM*: 64 MB
- *Festplatte:* 3,2 GB
- *Betriebsystem:* Windows 95

Server-Rechner 2 (HTTP-Server)

- *CPU*: Pentium 75 MHz
- *RAM*: 32 MB
- *Festplatte*: 450 MB
- *Betriebsystem*: Linux
- *HTTP-Server*: Apache

Client 1

- *CPU*: Pentium 60 MHz
- *RAM*: 32 MB
- *Festplatte*: 1,2 GB
- *Betriebsystem*: Windows 95
- *Browser*: Internet Explorer 4 und 5

Client 3

- *CPU*: Pentium III 450 MHz
- *RAM*: 64 MB
- *Festplatte*: 1,5 GB
- *Betriebsystem*: Windows 98
- *Browser*: Internet Explorer 5.0

Client 2

- *CPU*: Pentium 133 MHz
- *RAM*: 64 MB
- *Festplatte*: 3.2 GB
- *Betriebsystem*: Windows 95
- *Browser*: Internet Explorer 4 und 5

Der Testablauf :

1. Der zentrale OVID-Server und Dienst-Server werden gestartet
2. Konfigurieren des zentralen Servers (Eingeben des Start-URL, der beim Verbindungsaufbau an die Clients übergeben wird)
3. Client mit dem zentralen Benutzerinterface wird gestartet
4. Es erfolgt ein Verbindungsaufbau zum zentralen Server mit der anschliessenden Authentifizierung
5. Beim erfolgreichen Verbindungsaufbau schaltet der Client automatisch auf die Start-URL
6. Ab diesem Zeitpunkt ist die Synchronisationsphase abgeschlossen und der Benutzer kann auf die HTML-Dokumente mit den Vorlesungsinhalten zugreifen oder einzelne Dienste des Systems (Chat, Whiteboard, Audio-/Videokommunikation, Nachrichten, ftp und E-Mail) verwenden, falls sie durch die Ressourcenverwaltung freigeschaltet wurden
7. Schritte 3 bis 6 werden wiederholt, wobei verschiedene Rechner, Eingaben und Dienst-Kombinationen verwendet werden

In den durchgeführten mehrstündigen Testläufen funktionierte das System stabil (entsprechend dem oben aufgeführten geplanten Ablaufverhalten). Dabei wurde die Fähigkeit des Systems getestet, mehrere Benutzer und Arbeitsgruppen (bei Diensten, die die Gruppenarbeit unterstützen: Chat, Whiteboard) zu verwalten. Ausserdem wurde auch die Prozessorbelastung bei verschiedenen Nutzeranfragen/-aktivitäten sowohl auf dem Server [KUR00] (Tab. 6.1) als auch auf den Clients (Tab. 6.2) gemessen, der mittlere Fehler liegt bei ca. 0,5%.

Nutzeranzahl	Server 1 (600 MHz)	Server 2 (400 MHz)
1 bis 6	<1%	<1%
7	<1%	1%
8	1%	1%

Tabelle 6.1: CPU-Auslastung des Servers in Abhängigkeit von der Nutzeranzahl während einer Gruppenkommunikationssitzung [KUR00]

Dienste	Client 1 (60 MHz)	Client 2 (133 MHz)	Client 3 (450 MHz)
Benutzerinterface des zentralen Benutzermoduls	2%	1%	<1%
Verbindung zum Server hergestellt. Zugriff auf die Lehrinhalte ohne Audio-/Video-Daten	max. 40%	max 15%	max. 4%
Zugriff auf die Lehrinhalte mit Audio-/Video-Daten (MPEG)	100%, kein fliessender Bildablauf, CPU überlastet	90 bis 100%, fliessender Bildablauf (25 fps)	20 bis 30%, fliessender Bildablauf (25 fps)
Audio-/Videokommunikation zwischen zwei Teilnehmern	nicht möglich	80 bis 90%	20 bis 35%
Chat und Whiteboard	3%	2%	1%

Tabelle 6.2: CPU-Belastung des Clients bei verschiedenen Diensten

Dienste, die ohne Audio-/Videodaten auskommen, liessen sich auf jedem der Testrechner benutzen. Der mit 32 MB zu klein dimensionierter Hauptspeicher des ersten Clients hat zu hohen CPU-Belastung wegen Zugriffe auf eine Auslagerungsdatei geführt, das System blieb aber benutzbar. Beim abspielen von MPEG-Videos, die in die Testseite eingebettet wurden, kam es bei dem ersten Testrechner zur CPU-Überlastung. Das Video konnte nicht mehr fliessend (mit 25 fps) wiedergeben werden (Tab. 6.2). Bei dem zweiten Testrechner wurde beim Abspielen des Videos die CPU zwar stark belastet (bis zu 100%), der Bildaufbau blieb dabei aber fliessend bei 25 fps. Die grösseren Leistungsreserve als bei zwei ersten Rechnern zeigte der Testrechner Nr. 3. Die CPU-Belastung bei der Videodekodierung blieb bei max. 30%. Die Audio-/Videokommunikationskomponente für PAL-Qualität, die im Distance-Learning-System zum Einsatz kommt, funktionierte auf allen Testrechnern mit einer CPU ab 133 MHz, die mit den Videoschnittkarten ausgestatten waren, ohne Aussetzer bei der Audio- und Videoübertragung. Auf dem Client-Rechner mit einer 133 MHz–CPU lag die CPU-Belastung bei etwa 80 bis 90%, auf dem Rechner mit einem Intel PIII-450 MHz bei 20 bis 35%. Auf langsameren (weniger als 133 MHz) Rechnern ist der Einsatz dieser Komponente nicht vorgesehen. Dank der Tatsache, dass sowohl die Komprimierung als auch die

Dekomprimierung von Videodaten von jeweils einer separaten Videoschnittkarte durchgeführt wird, wird die CPU-Leistung nur für die Datenaufbereitung und Übertragung genutzt. Das ermöglichte eine Audio-/Videokommunikation in PAL-Qualität sogar auf einem Rechner der 133 MHz-Klasse.

Als kritisch wurden dabei die Festplatten-Zugriffszeit und Kapazität beim Web-Server und die Bandbreite seiner Netzanbindung bewertet:

Als Komprimierungsmethode für die Abspeicherung von Videos wurde die MPEG-Komprimierung gewählt. Im Unterschied zu einer Live-Übertragung spielt hier der Zeitverzug, der bei der Komprimierung und bei der Dekodierung entsteht, keine Rolle. Die Qualität ist mit ¼PAL und 25 Frames pro Sekunde für die Anforderungen des Systems optimal und der Datenstrom mit ca. 1,2 Mbit/s niedriger als bei Videodaten im AVI- und QuickTime-Format (bei vergleichbarer Qualität). Für eine Stunde Video im MPEG-Format werden ca. 600 Mbyte (bei einem Datenstrom von ca. 150 KByte/s, was einer Kompressionsrate von etwa 50:1 entspricht) benötigt. Ausserdem, wenn gleichzeitig mehrere Benutzer auf das System zugreifen, wovon auch auszugehen ist, müssen zur gleichen Zeit mehrere Videodateien ausgelesen werden. Beim gleichzeitigen Zugriff von 5 oder mehr Benutzern im Testlauf konnten die Daten nicht mehr schnell genug nachgeladen werden, so dass die Videosequenzen nur mit Pausen im Bildablauf angezeigt wurden. Deshalb sollte bei regulärem Betrieb ein Festplatten-Array, s.g. RAID-System (Redundant Array of Inexpensive Disks) verwendet werden, das gleichzeitiges Auslesen von mehreren Dateien ermöglicht. In RAID-Level 0 werden die Daten auf alle Laufwerke verteilt, ohne dabei Redundanz zu erzeugen. Die Kapazität errechnet sich aus der Summe der Einzelkapazitäten, und da die Daten auf mehrere Festplatten verteilt werden, können gleichzeitig mehrere Dateien ausgelesen werden.

Auch die Ansprüche an die Bandbreite, die dem Web-Server zur Verfügung steht, sind hoch: Schon dann, wenn das System nur von wenigen Anwendern (8 im Test) benutzt wird, steigt der Datenstrom auf mehr als 10 Mbit/s. Bei mehr als 8 Benutzern wird also die Bandbreite eines Ethernet-Anschlusses überschritten. Deshalb wird für den Web-Server ein ATM-Anschluss für den regulären Betrieb vorgeschlagen.

Wenn es bei dem Zugriff auf die abgespeicherten Vorlesungsinhalte zu Unterbrechungen kommt, bleibt das System trotzdem ohne Einschränkungen benutzbar. Wenn aber die Kommunikation zwischen dem zentralen Servermodul und den anderen Modulen wegen des vom Web-Server verursachten hohen Datenaufkommens gestört wird, reagiert das System langsamer auf die Aktivitäten des Benutzers und der Nutzungskomfort wird stark gemindert. Deshalb wird für den regulären Betrieb des Distance-Learning-Systems empfohlen, den Web-Server auf einem anderen Rechner, als dem, auf dem der zentrale Server läuft, zu installieren.

Beim Testen des Systems wurde ein Fehler im Internet Explorer 4.0 identifiziert, der in Darstellungsfehlern beim Zugriff auf die HTML-Inhalte resultiert. Das im Internet Explorer 4.0 verwendete Res-Protokol ist fehlerhaft implementiert. Der Microsoft Internet Explorer 4.0 ist aus diesem Grund für den Einsatz mit dem Distance-Learning-System OVID nur eingeschränkt geeignet. Nach dem Update auf die Version 5.0 waren diese Darstellungsfehler beseitigt. Daher wird für die Benutzung mit dem OVID der Microsoft Internet Explorer ab Version 5.0 empfohlen.

Die Netzanbindung des zentralen OVID-Servers ist weniger kritisch, als beim Web-Server [KUR00].

Das Distance-Learning-System OVID wurde auch für die Eignung zum Einsatz in schmalbandigen Netzwerken getestet.

Bei der Bandbreite von 115200 bps, die ungefähr der Bandbreite eines ISDN-Adapters mit zwei gebündelten Kanälen entspricht, reagiert der zentrale Server ohne Verzögerung auf die Anfragen des Clients. Nach dem erfolgten Einloggen in das System wurde eine Verbindung zum Web-Server aufgebaut. Die alphanumerischen Daten und die Testbilder wurden dabei auch verzögerungsfrei an den Client übermittelt. Beim Abspielen einer Test-MPG-Videosequenz mit einer Länge von ca. 6 Sekunden und einer Grösse von 1 MB kam es allerdings zu einer Wartezeit von länger als 1 Minute. Zwar wurde das Video bereits während des Ladevorgangs abgespielt, die Wiedergabe musste aber gestoppt werden, bis die ganze Datei geladen war, da es sonst zu mehrere Sekunden andauernden Aussetzern kam.

Die zweite Testreihe wurde bei einer Verbindungsbandbreite von 9600 bps durchgeführt.
Trotz dieser viel langsameren Verbindung kam es während der Kommunikation zwischen dem Client und dem zentralen Server zu keinen Verzögerungen Auf die Übertragung der Videosequenz wurde verzichtet, da schon bei dem ersten Test die Bandbreite als unzureichend bewertet wurde.

Somit ist eine Differenzierung zwischen der Übertragungsgeschwindigkeit eines ISDN-Adapters und noch langsameren Verbindungen unnötig. Die Ressourcenverwaltung stellt lediglich fest, ob die Bandbreite für eine Videoübertragung ausreicht. Aus einer einfachen Berechnung folgt, dass die benötigte Bandbreite dabei bei etwa 150KB/s*8=1,2 Mbit/s liegt. Mit 150KB/s ist der Datenstrom einer MPG-Videosequenz bei einer Kompressionsrate von ca. 50:1 und ¼-PAL-Qualität angegeben.

7 Zusammenfassung und Ausblick

Mit dieser Arbeit wurde ein Distance-Learning-System konzipiert und entwickelt, das eine modulare Architektur besitzt und für den Einsatz unter Windows 98/NT/2000 geeignet ist. Es ermöglicht einen komfortablen Zugriff auf die vorbereiteten und im Internet bereitgestellten Lehrmaterialien und ist dank seiner Architektur aufwärts kompatibel und skalierbar.

Das System besitzt folgende Eigenschaften:

- Unterstützung der Möglichkeiten der Netzwerke mit einer hohen Bandbreite
- Möglichkeit des Einsatzes in den schmalbandigen Netzwerken
- Standort des Benutzers irrelevant
- Automatische Ressourcenkontrolle
- Werkzeuge für die Gruppenarbeit
- Abstimmung der Systemteile aufeinander und ein leicht zu bedienendes Benutzerinterface
- modulare Struktur, offene Architektur

Mit dem Dienst zur Wiedergabe von Vorlesungen wird dem Benutzer eine Möglichkeit bereitgestellt, auf die Vorlesungsinhalte zuzugreifen. Dafür wurde ein eigener Web-Browser (basierend auf dem Browserkern von Microsoft Internet Explorer) entwickelt, der die Möglichkeit zum Starten von Zusatzdiensten besitzt. Dieser Browser dient dem Benutzer als eine Schaltzentrale und übernimmt die Funktionen einer zentralen Anwenderapplikation. Bei den Lehrinhalten werden die Möglichkeiten der Breitbandnetzwerke für die Audio-/Videowiedergabe genutzt. Es stehen mehrere Gruppenarbeit-Werkzeuge für die Gruppenarbeit zur Auswahl. Eines dieser Werkzeuge, ein Dienst zur Audio-/Videokommunikation, ermöglicht eine komfortable Kommunikation zwischen zwei Benutzern. Alle Synchronisations- und Steuervorgänge laufen automatisch ab. Das OVID stellt somit ein integriertes System dar, dessen Teile aufeinander abgestimmt sind. Aus diesem Grunde gestaltet sich die Nutzung des Systems sehr einfach, es kann auch von einem ungeschulten Benutzer gesteuert werden. Für die Speicherung der Benutzerdaten wurden im Rahmen dieser Arbeit eine Benutzerdatenbank implementiert und die Benutzerverwaltung realisiert.

Das System wurde erfolgreich getestet (Kap. 6). Es wurde die Fähigkeit des Systems analysiert, mehrere Benutzer und Arbeitsgruppen zu verwalten. Ausserdem wurde auch die Prozessorbelastung bei verschiedenen Nutzeranfragen/-aktivitäten gemessen. Die Teilkomponenten wurden auf ihre Eignung zum Einsatz mit verschiedener Hardware überprüft.

Obwohl alle in dieser Arbeit angestrebten Ziele erreicht wurden, gibt es noch einige Richtungen, in die die Weiterentwicklung des Distance-Learning-Systems gehen kann. So kann zum Beispiel ein Tool entwickelt werden, um die Eingabe der Lehrinhalten in Form von HTML-Dokumenten zu vereinfachen und zu automatisieren. Des weiteren kann noch ein sehr wichtiger Dienst zur Durchführung von virtuellen Veranstaltungen implementiert werden. Bei einem solchen Dienst wird der Vortragende in Echtzeit mit einer Videokamera aufgenommen und der Video- und Audiostrom wird synchron mit den dazu passenden Folieninhalten an die Teilnehmer einer virtuellen Veranstaltung übertragen. Um die Videoübertragung zu realisieren kann auch der bereits existierende Audio-/Video-Kommunikationssystem verwendet werden, das Sitzungsmanagement soll dazu entsprechend modifiziert werden.

Anhang A: TCP/IP – Referenzmodell

Das TCP/IP-Referenzmodell wurde zunächst im APRANET angewendet, aus dem sich später das Internet entwickelt hat. Die Referenzarchitektur sieht vor, dass mehrere heterogene Netze nahtlos miteinander zusammengeschlossen werden. Des weiteren soll das Netz gegenüber den Ausfällen von Teilnetzen unempfindlich sein, d.h. die Verbindungen sollen intakt bleiben, solange die Quell- und Zielmaschinen funktionieren. Deshalb wurde es als ein paketvermitteltes Netz auf der Grundlage einer verbindungslosen Vernetzungsschicht aufgebaut [TAN98].

TCP/IP basiert auf einem 3-Schichten-Modell:

Die Internet-Schicht definiert ein offizielles Packetformat und Protokoll namens IP (Internet Protocol). Sie hat die Aufgabe, IP-Pakete richtig zuzustellen. Die wichtigste Frage ist dabei das Packet-Routing, ebenso wie das Vermeiden von Überlastungen. Der Aufbau des Headers eines IP-Datengramms wird in der Abb. A.A.1 dargestellt.

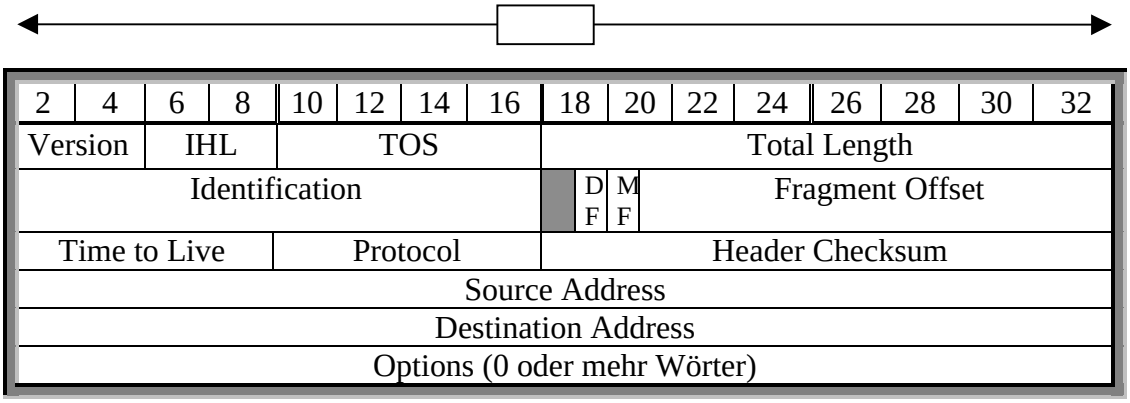

Abb. A.A.1: Der Header des IP-Protokolls

Bedeutung einzelner Felder des Headers:

- *Version* – Version des Protokolls

- *IHL* – Header-Länge in 32-Bit-Wörtern
- *TOS (Type of service)* – der gewünschte Dienst
- *Total Length* – Länge des kompletten Datagramms bis max. 65.535 Byte
- *Identification* – erforderlich, um die Zugehörigkeit eines neu angekommenes Fragments zu einem Datagramm festzustellen
- *DF (Don`t Fragment)* – Befehl an die Router, wenn ein Datagramm nicht fragmentiert werden darf
- *MF (More Fragments)* – bei allen Fragmenten ausser dem letzten gesetzt
- *Fragment Offset* – bezeichnet, an welche Stelle im Datagramm ein Fragment gehört
- *Time to Live* – Zähler, mit dem die Lebensdauer von Paketen begrenzt werden kann
- *Protocol* – bestimmt, an welchen Transportprozeß ein Datagramm weiterzugeben ist
- *Header Checksum* – Prüfsumme des Headers
- *Source Address* und *Destination* Address – bezeichnen die Netz- und Hostnummer

Jeder Host und Router besitzt im Internet eine eindeutige, 32 Bit lange IP-Adresse, die die Netz- und Hostnummer kodiert. Die für die IP-Adresse benutzten Formate sind in Abb. A.A.2 dargestellt.

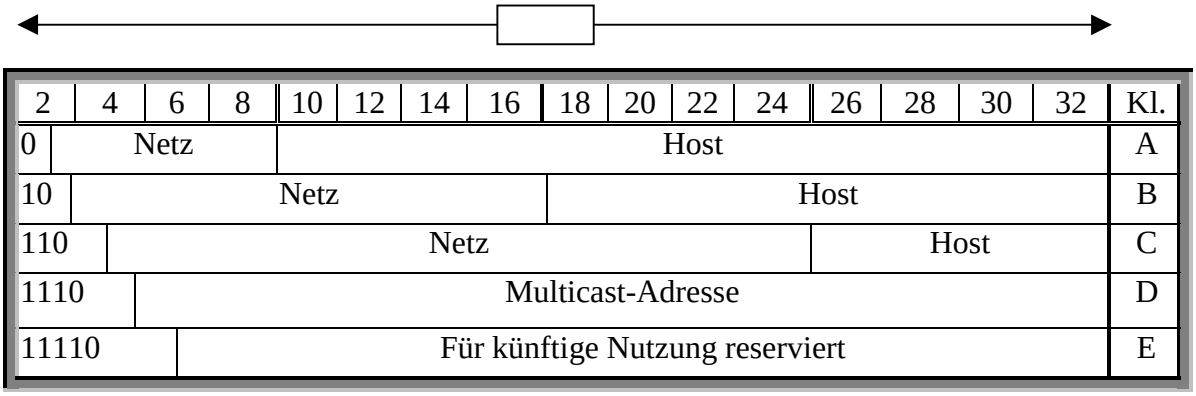

Abb. A.A.2: IP-Adreßformate

Die Formate der Klasse A, B, C und D erlauben die Konfiguration von bis zu 126 Netzen mit je 16 Millionen Hosts, 16.382 Netzen mit bis zu 64.000 Hosts, 2 Millionen Netzen mit bis zu je 254 Hosts und Multicast, durch das ein Datengramm an mehrere Teilnehmer gesendet werden kann.

Die Adreßbereiche der Klassen A, B, C, D und E sind in der Tabelle A.A.1 dargestellt.

Klasse	Adreßbereich
A	1.0.0.0 bis 127.255.255.255
B	128.0.0.0 bis 191.255.255.255
C	192.0.0.0 bis 223.255.255.255
D	224.0.0.0 bis 239.255.255.255
E	240.0.0.0 bis 247.255.255.255

Tab. A.A.1 Adreßbereiche

Die übliche IP-Kommunikatin findet zwischen einem Sender und einem Empfänger statt. Bei manchen Anwendungen ist es aber notwendig, einem Prozess die Übertragung an viele Empfänger gleichzeitig zu ermöglichen, z.B. im Rahmen verteilter Datenbanken, aktuellen Börsennotierungen, digitale Konferenzen usw.
IP unterstützt Multicasting durch Verwendung von Adressen der Klasse D. Jede Adresse der Klasse D identifiziert eine Hostgruppe. Zu Identifizierung von Gruppen sind 28 Bit verfügbar, so dass gleichzeitig über 250 Millionen Gruppen bestehen können. Sendet ein Prozess ein Paket an eine Adresse der Klasse D, wird es an alle Mitglieder der adressierten Gruppe unter besten Bemühungen, aber ohne Garantien zugestellt.
Es werden zwei Arten von Gruppenadressen unterstützt: permanente und temporäre. Eine permanente Gruppe besteht immer und muss nicht eingerichtet werden. Jede permanente Gruppe hat eine permanente Gruppenadresse. Hier sind zwei Beispiele von permanenten Gruppenadressen:

224.0.0.1 Alle Systeme in einem LAN
224.0.0.2 Alle Router in einem LAN

Temporäre Gruppen müssen vor der Nutzung erstellt werden. Ein Prozess kann seinen Host um den Beitritt zu einer bestimmten Gruppe und um Austritt aus einer Gruppe anfragen. Verlässt der letzte Prozess eines Hosts eine Gruppe, ist diese Gruppe nicht mehr am Host präsent. Jeder Host verwaltet die Prozesse, die momentan an einer Gruppe teilnehmen.
Multicasting wird durch spezielle Multicast-Router implementiert, die mit Standard-Routern koexistieren können. Etwa einmal pro Minute sendet jeder Multicast-Router ein Hardware-Multicast an die Hosts in seinem LAN und fordert sie auf, die Gruppen zu melden, an denen

ihre Prozesse momentan teilnehmen. Jeder Host sendet seine Antwort für alle Adressen der Klasse D zurück, an denen er interessiert ist.

Die Transportschicht ist eine Schicht oberhalb der Internet-Schicht. Ihre Aufgabe ist es, den Partnereinheiten an den Quell- und Zielhosts die Kommunikation zu ermöglichen. Dazu wurden zwei Ende-zu-Ende-Protokolle definiert:

1. Das **TCP** (Transmission Control Protocol) ist ein zuverlässiges verbindungsorientiertes Protokoll, durch das ein Bytestrom von einer Maschine fehlerfrei im Internet einer anderen Maschine zugestellt wird. Der eingehende Bytestrom wird dabei in einzelne Nachrichten zerlegt, die an die Internet-Schicht weitergeleitet werden. Es enthält auch die Flußteuerung für die Synchronisation zwischen Sender und Empfänger. TCP ist formell in RFC 793 definiert. Die Klärungen und einige Fehlerbehebungen von Inkonsistenzen, die man im Laufe der Zeit gefunden hat, sind ausführlich in RFC 1122 beschrieben. Erweiterungen sind in RFC 1323 definiert.

   Um einen TCP-Dienst nutzen zu können, müssen Sender und Emfänger Endpunkte – sogenannte Sockets – erstellen, siehe Kapitel ***Sockets***.

   Bei der Kommunikation tauschen die sendenden und die empfangenden TCP-Einheiten Daten in Form von Segmenten aus. Ein Segment besteht aus einem festen 20-Byte-Header (Abb. A.A.3), einem optionalen Teil und den Datenbytes. Die Grösse von Segmenten wird von der TCP-Software festgelegt.

   Hier ist die Bedeutung einzelner Felder des Headers:

   - *Quellport* und *Zielport* – identifizieren die lokalen Endpunkte der Verbindung
   - *Folgenummer* – gibt das als nächstes erwartete Byte an
   - *TCP-Header-Länge* – gibt an, wie viele 32-Bit-Wörter im TCP-Header enthalten sind
   - 1-Bit-Flags:
     - *URG* – wird auf 1 gesetzt, wenn Urgent Pointer benutzt wird
     - *ACK* – wird auf 1 gesetzt, um anzugeben, dass die Bestätigungsnummer gültig ist
     - *PSH* – bezeichnet PUSH-Daten

- *RST* – wird benutzt, um eine Verbindung zurückzusetzen, wenn ein Host abgestürzt oder eine ähnliche Störung aufgetreten ist
- *SYN* – wird benutzt, um Verbindungen aufzubauen
- *FIN* – bestimmt, dass der Sender keine weiteren Daten mehr zu übertragen hat

- *Zeitfenstergrösse* – bezeichnet, wie viele Bytes ab dem bestätigten Byte gesendet werden können
- *Prüfsumme* – prüft den Header, die Daten und den konzeptionellen Pseudo-Header, um hohe Zuverlässigkeit zu bieten
- *Optionen* – bieten eine Möglichkeit, zusätzliche Funktionen bereitzustellen, die sonst im Header nicht verfügbar sind
- *Daten* – bis zu 65.535-20-20=65.515 Datenbyte, wobei sich die ersten 20 auf den IP-Header und die zweiten 20 auf den TCP-Header beziehen

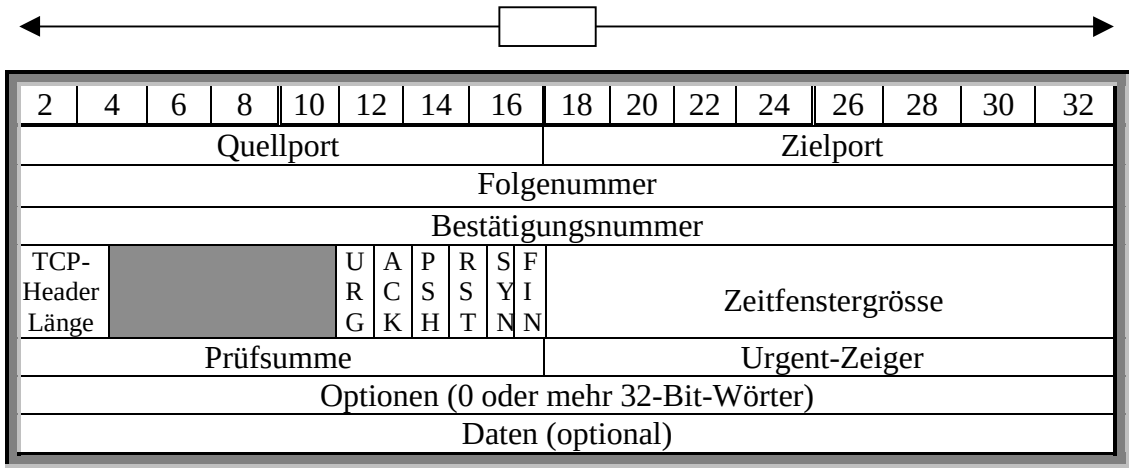

Abb. A.A.3: Der TCP-Header

2. Das **UDP** (User Datagram Protocol) ist ein unzuverlässiges verbindungsloses Protokoll für Anwendungen, die anstelle der Abfolge oder Flußkontrolle von TCP diese Aufgaben selbst bereitstellen. Es bietet eine Möglichkeit für Anwendungen, gekapselte rohe IP-Datengramme zu übertragen, ohne eine Verbindung aufzubauen. UDP ist in RFC 768 definiert.

   Ein UDP-Segment besteht aus einem 8-Byte-Header, gefolgt von den Daten. Der Header ist in der Abb. A.A.4 dargestellt.

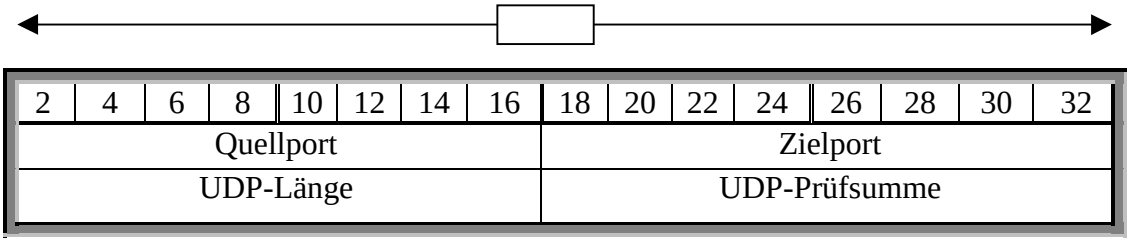

Abb. A.A.4: Der UDP-Header

Die Verarbeitungsschicht befindet sich über der Transportschicht und umfaßt alle höherschichtigen Protokolle, zu dennen z.B. TELNET, FTP oder SMTP zählen. Im Laufe der Jahre wurden dazu viele andere Protokolle hinzugefügt, wie z.B. DNS, NNTP und HTTP, das Protokoll des World Wide Webs.

Anhang B: Sockets

Sockets wurden zum esten Mal im Jahre 1981 mit Unix BSD 4.2 vorgestellt und dienten als ein Interface, das die Unix-to-Unix Kommunikation sicherstellen sollte. Heute werden die Sockets von praktisch allen Betriebsystemen unterstützt. Die Windows-Socket-API, auch als WinSock bekannt, stellt eine vielschichtige Spezifikation dar, die den Gebrauch von TCP/IP unter Windows-Betriebsystemen standardisiert. Die WinSock API basiert auf dem Berkeley Sockets Interfase. In dem BSD Unix Betriebsystem sind die Sockets ein Bestandteil des Kernels, auf Windows-Systemen werden sie in Form der Bibliotheken vertrieben. Man kann sagen, dass die Sockets heutzutage ein de facto portabler Standart für die Netzwerkapplikationen sind, die auf TCP/IP Netzwerke aufsetzen.

Ein Socket repräsentiert einen logischen Kommunikationskanal zwischen zwei Prozessen. Er kann auch bidirektional sein. Sowohl der lokale als auch der externe Prozess müssen ein gemeinsames Netzwerkprotokoll (z.B. TCP für verbindungsorientierte Interprozesskommunikation) wählen.

Es existieren drei Arten von Sockets – Stream, Datagramm und Raw. Stream- und Datagrammsockets benutzen das TCP bzw. UDP-Protokoll, Rawsockets haben ein direktes Interface zum IP-Protokoll. Die Art der Sockets wird zum Zeitpunkt ihrer Initialisierung spezifiziert.

Die Adresse eines Sockets besteht aus zwei Teilen: aus einer IP-Adresse und der Nummer des Ports. Wie eine IP-Adresse aufgebaut ist, wurde bereits ausführlich im Kapitel ***TCP/IP – Referenzmodell*** besprochen. An dieser Stelle ist es notwendig zu erklären, was sich hinter dem Begriff ‚Port' verbirgt.

Ein **Port** ist ein Eingangspunkt in eine Application, die auf einem Host läuft. Es wird mit einer 16-bit langen Integerzahl repräsentiert. Die Ports werden normalerweise für die Definition von Services verwendet, die von verschiedenen Serverapplikationen zur Verfügung gestellt werden. Allgemein nennt man die Portnummern unter 256 gut bekannte Ports (well-known Ports). Sie sind für Standarddienste reserviert. Jeder Prozess, der z.B. mit einem Host eine Verbindung aufbaut, um über FTP eine Datei zu übertragen, kann sich an Port 21 des Zielhosts anklinken, um dessen FTP-Daemon anzusprechen. Auf ähnliche Weise wird Port 23

für eine entfernte Login-Sitzung unter Telnet benutzt. Die Liste der gut bekannten Ports ist in der RFC 1700 enthalten.

Ein Socket kann für mehrere Verbindungen gleichzeitig benutzt werden. Anders ausgedrückt, zwei oder mehr Verbindungen enden auf dem gleichen Socket. Verbindungen werden nach Socketbezeichner an beiden Enden identifiziert, d.h. (socket1, socket2).Virtuelle Verbindungsnummern oder andere Bezeichner werden nicht benutzt.

Streamsockets benutzen die verbindungsorientierte Art der Kommunikation. Das heisst, eine Verbindung wird zwischen den zwei Rechnern zum Sitzungssanfang aufgebaut, und sie besteht während der gesamten Session. Jedes Informationspaket, das über diese Verbindung ausgetauscht wird, erhält seine individuelle Nummer, was das Verfolgen von einzelnen Paketen im Netz und Bestätigen der Fehlerfreiheit der Übertragung ermöglicht. Die duplizierten Pakete werden dabei aufgespürt und eliminiert. Der Preis, den man für diese Zuverlässigkeit der Übertragung bezahlen muss, ist ein grosser Mehraufwand für die Initialisierung und das Management einer Sitzung. Wenn die Verbindung aus irgendeinem Grund gekappt wird, muss eine der an der Kommunikation beteiligten Seiten die Verbindung erneut aufbauen.

Datagramme – auch bekannt als verbindungslose Protokolle oder ‚transmit and pray' Protokolle – stellen eine einfache aber unzuverlässige Form der Kommunikation dar. Sie sind in dem Sinne unzuverlässig, dass sie nicht verfolgt oder bestätigt werden. Man sendet ein Datagramm und hofft dabei, dass es auf der anderen Seite ankommt. Das typische Einsatzfeld der Datagramme sind kurze, kontroll-artige Mitteilungen, z.B. regelmässig gesendete Nachrichten an das Sitzungsmanagement, dass man noch an der Kommunikation teilnimmt.

Windows Sockets

WinSock wurde als Ergebnis einer Empfehlung eines Firmenkonsortiums unter Führung von Microsoft entwickelt. Es vereinheitlicht die Anwendungs-Programmierschnittstellen und definiert eine binäre Schnittstelle, an die sich alle WinSock-Hersteller halten müssen. Die Spezifikationen definieren die Bibliotheksaufrufe und die damit verknüpfte Semantik, die von Netzwerksoftware-Herstellern implementiert und von Anwendungsprogrammierern benutzt wird. Das Ergebnis ist, dass Anwendungen, die für WinSock geschrieben werden, mit den Bibliotheken verschiedener Hersteller und mit den Protokollen beliebiger WinSock-

kompatibler Hersteller (aufgrund der binären Schnittstelle) zusammenarbeiten. Mit der Version 4.0 von Windows NT wurde von Microsoft die Programmierschnittstelle Windows Sockets in der Version 2.0 (WinSock 2) ausgeliefert. Mit WinSock 2 erhält man eine neue Flexibilität, denn die neue DLL mit dem Namen WS2_32.DLL muss nicht mehr von jedem Hersteller auf den eigenen Protokollstack angepasst werden, sondern unterstützt ein standardisiertes Service Provider Interface (SPI). In der Abb. A.B.1 ist die Architektur von WinSock2 dargestellt.

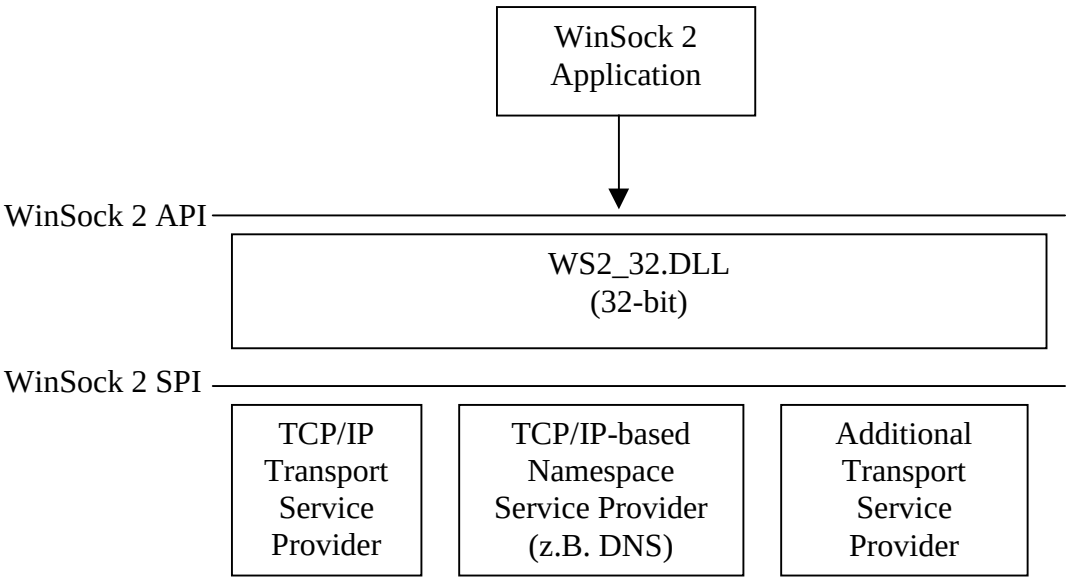

Abb. A.B.1: WinSock 2 - Architektur

Der Grund, warum WinSock auch andere Protokolle als TCP/IP unterstützt, ist der, dass Windows Sockets zwar als Schnittstelle im Berkley-Stil für TCP/IP für Windows konzipiert wurde, sich aber rasch als Standard für die Netzwerkprogrammierung über mehrere Protokolle in dieser Umgebung etabilierte.

Microsoft fordert die Entwickler von Netzwerksoftware dazu auf, WinSock-Schnittstellen zu schreiben, anstatt auf individuelle Low-Level-Schnittstellen zuzugreifen. Dieses Modell einer einzelnen standardisierten Schnittstelle bietet zwei wesentliche Vorzüge:

- Eine Client/Server-Anwendung kann auf jedes beliebige zugrundeliegende Protokoll über WinSock zugreifen.
- Ein Client/Server-Design kann eine einzelne Kommunikationsbibliothek (API, C++-Klassen etc.) entwickeln, die mit mehreren Protokollen wie AppleTalk oder IPX/SPX arbeitet. Mit WinSock besteht ein grosses Potential zur Code-Reduktion.

Die Microsoft Foundation Class Library (MFC) unterstützt die Programmierung der Windows Socket API durch zwei Klassen.

Die eine Klasse, CAsyncSocket, stellt eine Inkapsulation der Windows Socket API dar. Diese Klasse macht nichts anderes, als die Sockets in einer objektorientierten Form für den Gebrauch in C++ bereitzustellen.

Die andere Klasse, CSocket, liefert einen hohen Abstraktionsgrad, um die Programmierung der Netzwerkkommunikation zu vereinfachen. Sie erlaubt es, mit den Sockets in Verbindung mit den Objekten der CArchive-Klasse zu arbeiten. Die Klasse CSocket unterstützt alle Funktionen der Klasse CAsyncSocket. Ausserdem werden von dieser Klasse viele Aspekte der Kommunikation übernommen, die bei der Benutzung der CAsyncSocket vom Programmierer selbst erledigt werden müssten, z.B. das Blocking.

Die Sockets können in zwei Modi angesprochen werden, in dem Blocking- und Nicht-Blockingmodus. Diese Modi werden auch Synchron- oder Asynchronmodus genannt.

1. In dem Blocking-Modus blockiert das Socket das gesamte System, bis die Übertragung abgeschlossen ist.
2. Beim Asynchronmodus wird die Kontrolle gleich nach dem Aufruf der Funktion an das Hauptprogramm zurückgegeben.

Beim Asynchronmodus muss das Programm davon in Kenntnis gesetzt werden, wann eine Operation abgeschlossen ist. Zu diesem Zweck bietet die Klasse CAsyncSocket virtuelle Rückruffunktionen, welche von der Anwendung bereitgestellt und vom System aufgerufen werden können. Mit Hilfe der Funktion AsyncSelect wird festgelegt, über welche Ereignisse eine Benachrichtigung erfolgen soll. Die Tabelle A.B.1 zeigt die möglichen Ereignisse.

Ereignis	Konstante
Lesebereitschaft	FD_READ
Schreibbereitschaft	FD_WRITE
Ankunft von ‚Out-of-Band'-Daten	FD_OOB
Eingehende Verbindung	FD_ACCEPT
Zustande gekommene Verbindung	FD_CONNECT
Schliessen des Sockets	FD_CLOSE

Tab. A.B.1: Socket-Ereignisse

In der Literatur wird Empfohlen, die asynchronen Sockets einzusetzen, denn sonst kann der normale Ablauf des Programms durch die während der Kommunikation auftretenden Verzögerungen gestört werden. Allerdings erfordert der Einsatz von Sockets, die im Asynchronmodus betrieben werden, eine aufwendige Synchronisaton. Diese Synchronisation ist in der Klasse CSocket bereits vollständig implementiert.

Anhang C: HTTP

Das *Hypertext Transfer Protocol* (HTTP) wird im Internet erst seit 1990 verwendet. Die erste Version des HTTPs – HTTP/0.9 – war ein einfaches Protokoll für die Übertragung von Raw-Daten über das Internet. HTTP/1.0, was im Jahre 1994 eingeführt war, stellte die selbstbeschreibende Nachrichten, die auf MIME-Kodierung basierten, vor. 1998 wurde von dem W3C HTTP/1.1 verabschiedet, eine Version, die zu der Version 1.0 kompatiebel blieb und effektivere Wege der Nutzung von TCP einführte.

HTTP ist ein RPC-(Remote Procedure Call) Protokoll, das auf TCP/IP aufsetzt. Es wird für den Zugriff auf die URL-Ressourcen benutzt. Der Client richtet als erstes eine Verbindung zum entfernten Rechner ein, und dann wird an den Server ein Request übergeben. Der Server bearbeitet diesen Request, liefert seine Antwort zurück und beendet die Verbindung.

Der Client – typisch ein Web-Browser – fordert beim Server eine Hypertext-Seite an, und dann werden für alle Bild- oder Audiodateien, die in diesem Dokument eingebettet sind, separate Requests geschickt. Dabei werden beim HTTP 1.0 für jede einzelne dieser Dateien neue Verbindungen aufgebaut. Dieser Weg war nicht sehr effizient, und deshalb wurden mit der Einführung von HTTP 1.1 drei neue Mechanismen entwickelt, um die Übertragung von Dokumenten effizienter zu machen:

- *Persistent Connections* erlauben, während einer Vebindung gleich mehrere Dateien anzufordern, eher die Verbindung geschlossen wird. Entweder der Client oder der Server kann die Verbindung erst dann, wenn es notwendig ist, trennen. Persistänte Verbindungen sind die Default-Verbindungen im HTTP 1.1

- *Pipelining* bietet dem Client eine Möglichkeit, multiple Anfragen abzuschicken, ohne dabei auf die Antwort des Servers zu warten. Der Server muss dann diese Anfragen in derselben Reihenfolge beantworten, in der sie angekommen sind. Der Client soll ausserdem dazu bereit sein, seine Anfragen an den Server erneut zu senden, falls die Verbindung vom Server getrennt wurde.

- *Cache Validation Commands* reduzieren die Netzwerkbelastung und machen das Protokoll effektiver. HTTP 1.1 bietet mehrere Befehle an, die dem Client helfen

sollen, die Konsistenz des lokalen Caches, in dem Dokumente abgespeichert werden, sicherzustellen.

Somit liegt der grösste Unterschied zwischen den HTTP 1.0 und HTTP 1.1 in der Tatsache, dass beim HTTP 1.1 eine Verbindung, einmal aufgebaut, für einen längeren Zeitraum bestehen bleibt, und dabei die Antwortzeiten verkürzt und die Netzbelastung verringert werden.

Eine wichtige Eigenschaft von HTTP ist die Fähigkeit, selbstbeschreibende Daten unter Verwendung von MIME verschicken zu können. Das Protokoll erlaubt es dem Browser, den Server über alle Datentypen zu informieren, die der Browser darstellen kann. Das muss immer beim Aufbau einer Verbindung zwischen dem Server und dem Client gemacht werden, denn die HTTP Server sind s.g. Zustandslose Server, die keine Protokolle über ihre Verbindungen führen oder abspeichern, und somit über die Eigenschaften eines Browsers bei jedem Verbindungsaufbau neu informiert werden müssen. Dies erlaubt einerseits eine effiziente Funktionsweise eines HTTP-Servers, andererseits kann es die Implementierung von komplexeren Web-orientierten Systemen erschweren. Um diese Mängel des HTTP-Protkolls zu kompensieren, wurden viele zusätzliche Mechanismen wie z.B. Cookies entwickelt, deren vollständige Beschreibung den Rahmen dieser Arbeit sprängen würde.

HTTP hat vieles mit der *Internet Mail* [RFC 822] und den *Multipurpose Internet Mail Extensions* (MIME) [RFC 1521] gemeinsam, die erweiterbare Mechanismen zum Austausch von Multimedia-E-Mails bereitstellen. MIME erweitert RFC 822, indem sie zusätzliche Headers spezifiziert, die den Inhalt der Nachricht beschreiben. Zum Beispiel definiert MIME Headers, die spezifizieren, ob eine Nachricht aus mehreren Teilen zusammengesetzt ist, und wie diese Teile dann getrennt sind. Ein MIME-Header benutzt das Format ‚content-type: type/subtype', z.B. ‚content-type: text/html'. Es werden sieben Datentypen unterstützt, darunter Text; Audio; Video; Bilder; Nachrichten, die auf eine andere Nachricht zeigen können; Multipart-Nachrichten, bei dennen alle Teilnachrichten vom verschiedenen Typ sein können; und Applikation-spezifische Daten. Jeder MIME-Typ kann auch über mehrere Subtypen verfügen.

Die Content-Type-Header von HTTP sind den Headern von MIME sehr ähnlich. Die Entwickler von HTTP haben aber entschieden, sich nicht an die Regeln, die von RFC 822 und

MIME vorgeschlagen wurden, zu halten. Somit ist HTTP keine MIME-kompatible Applikation.

Anhang D: CGI

Häufig werden Webseiten gebraucht, die nicht nur statische Informationen tragen. Diese Seiten werden mit Hilfe von sogenannten CGI-(Common Gateway Interface-)Programmen realisiert, die üblicherweise mit Skriptsprachen wie Perl realisiert sind.

Erhält ein HTTP-Server einen Request, kann der spezifizierte URL nicht nur auf eine HTML-Seite verweisen, sondern auch auf ein lauffähiges Programm des Servers zeigen. In diesem Fall sendet der Server nicht einfach den Inhalt eines Dokuments an den Client zurück, sondern führt das spezifizierte Programm aus. Dessen Aufgabe ist es dann, eine Ausgabe zu erzeugen, die der Server entgegennimmt und dem Client, also dem Browser, so zurückliefert, als wäre es eine Webseite.

Dabei kann das CGI-Programm seine Informationen aus beliebigen Quellen beziehen: Aus einer Datenbank, aus einer Datei des Servers oder sogar von einem anderen Rechner. Den HTTP-Server interessiert nur die Ausgabe des CGI_Programms, die dieses üblicherweise mit HTML strukturiert.

Der Aufbau einer dynamischen Webseite kann vom CGI-Programm auch nach speziellen Wünschen des Clients abgewickelt werden. Den hierzu notwendigen Kommunikationskanal öffnet die Formulartechnik.

Formulare sind Web-Seiten, die eine Reihe von Feldern beinhalten, die auf Benutzereingaben reagieren. Editierbare Textfelder, Auswahlboxen und verschiedenartige Buttons werden dazu genutzt, individuelle Anfragen zu starten. Die wichtigsten dieser Formulartypen sind:

- Editierbare Formularfelder
- Check- und Radiobuttons
- Listboxen
- Submit- und Reset-Buttons

Werden vom Benutzer die Formularfelder ausgefüllt und der Submit-Button gedrückt, reagiert der Server spezifisch auf die Eingabe und sendet eine entsprechende Antwort. Dabei wird bei einem Klick auf den Submit-Button ein HTTP-Request an den Server abgesetzt, der den

editierten Inhalt der Textfelder gleich mitliefert. Der im CGI-Formular enthaltene URL spricht auf der Serverseite ein CGI-Programm an, das diese Parameter analysiert, entsprechend reagiert und eine Ausgabe produziert. Diese wird von dem HTTP-Server wie ein HTML-Dokument zurück an den Client geliefert.

Die Einstellungen am Formular, die der Benutzer vornimmt, können über zwei verschiedene Verfahren zum Server gelangen. Die Information gedrückter Check- oder Radio-Buttons sowie ausgefüllter Textfelder codiert der Browser in ein Name-Value-Schema und sendet entweder

- einen GET-Request oder
- einen POST-Request

an das CGI-Programm. Der Unterschied zwischen beiden Verfahren besteht darin, dass der GET-Request die Name-Value-Paare der Formularinformation in einem codierten Format an den URL anhängt, so dass sie vom Server interpretiert und dem anlaufenden CGI-Programm in einer Environment –Variable übermittlet werden. Die Parameter eines POST-Requests sind hingegen nicht im URL sichtbar, sondern gelangen gesondert zum Server, der diese dem zuständigen CGI-Programm in den Standard-Input überspielt.

Die POST-Methode bietet sich für grössere Formulare an, da ein URL mit angehängter Zusatzinformation schnell unhandlich und unleserlich wird und ausserdem einer Längenbeschränkung von 1024 Bytes unterliegt.

Anhang E: COM

1990 stellte Microsoft die *Object Linking and Embedding 1* (OLE 1) Technologie als seine grundlegende Strategie vor, um multiple Applikationen und Multimedia-Datentypen im Rahmen eines zusammengesetzten Dokuments vereinen zu können. OLE 1 war ein fehlerbehaftetes und langsames Protokoll, das auf DDE (Dynamic Data Exchange) aufsetzte. Wenn eine Applikation aus einem zusammengesetzten Dokument gestartet wurde, enthielt sie ihr eigenes Fenster, und die benötigten Daten wurden in ihr Adressenraum kopiert. Es gab keinen Weg, wie diese Applikation direkt auf das Dokument zugreifen könnte.

OLE 2, das im Jahre 1993 eingeführt wurde, bereinigte viele dieser Mängel mit einer neuen objektorientierten Technologie namens *Component Object Model* (COM). Das OLE 2 setzt vollständig auf COM auf.

Im Frühjahr 1996 hat Microsoft *ActiveX* vorgestellt, was nichts anderes als OLE-Objekte für den Einsatz im Internet darstellt. Noch im selben Jahr wurde auch die erste Version des verteilten COMs – DCOM – als ein Teil des Windows NT 4.0 verabschiedet.

COM definiert ein Kommunikationsprotokoll für Programmkomponenten. *COM Controls* sind wiederverwendbare Komponenten auf binärer Ebene, die sprachunabhängig sind. Sie werden als COM Mini-Server implementiert. Deshalb ist ein COM Control keine eigenständige Anwendung, sondern muss in einem COM-Container eingebunden werden. Die Abb. A.E.1 zeigt Architektur von COM Controls.

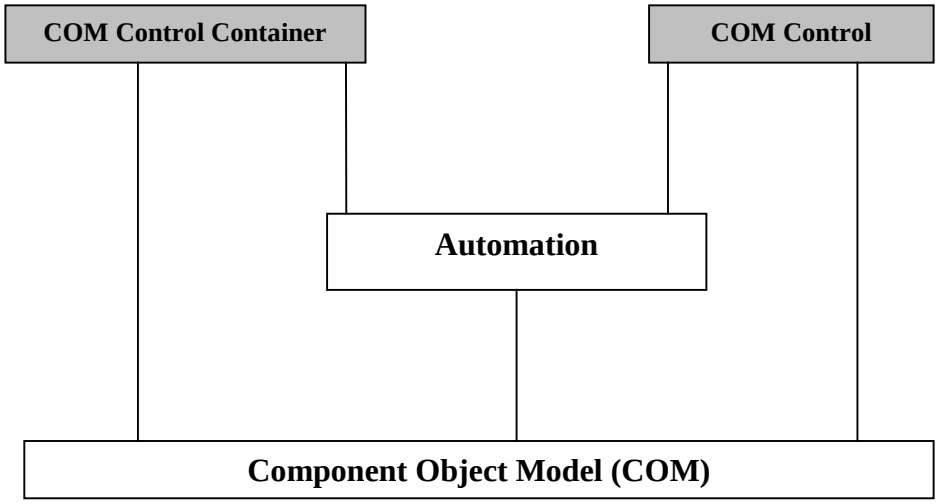

Abb. A.E.1: Architektur von COM Controls

COM Objekte – auch als ActiveX Objekte bekannt – sind Komponenten, die ein oder mehrere Interfaces unterstützen, wie es die Klasse des Objekts definiert. Diese Komponenten zeichnen sich durch ihre Abgeschlossenheit und die Einbettung in einem Container aus. Sie haben grundsätzlich Eigenschaften (*Properties*), bieten Dienste (*Methoden*) an und senden Ereignisse (*Events*) an den Container zurück. Bei Komponenten im Umfeld einer grafischen Oberfläche kommt noch eine grafische Präsentation hinzu (Abb. A.E.2). Controls und Control-Container kommunizieren über definierte Schnittstellen.

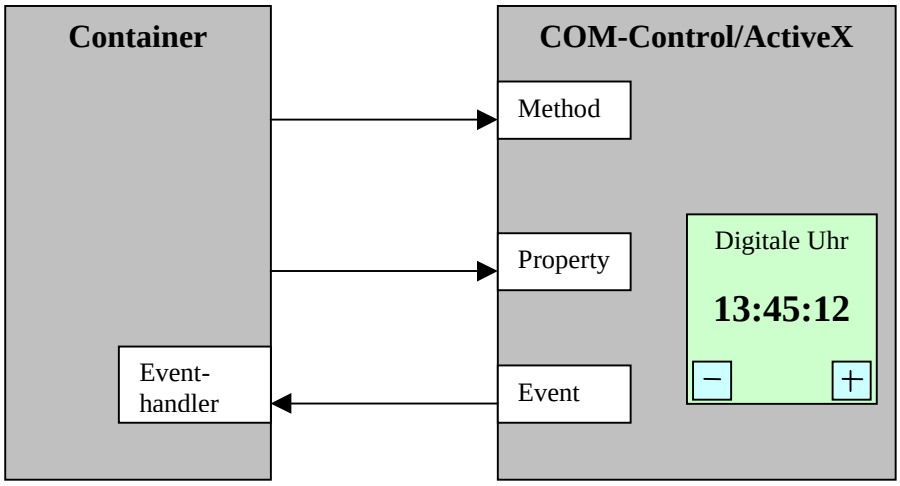

Abb. A.E.2: Kommunikation zwischen COM Control und COM Control-Container

Abkürzungsverzeichnis

ATM	Asynchronous Transfer Mode
CGI	Common Gateway Interface
CCCP	Conference Control Channel Protocol
COM	Component Object Model
CSCW	Computer-Supported Collaborative Work
CSS	Cascading Style Sheets
DAO	Data Access Objects
FTP	File Transfer Protocol
GUI	Graphical User Interface
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
ICMP	Internet Control Message Protocol
IETF	Internet Engineering Task Force
IP	Internet Protocol
ISDN	Integrated Services Digital Network
JaTeK	Java based Teleteaching Kit
JDK	Java Development Kit
JVM	Java Virtual Mashine
LAN	Local Area Network
LDAP	Lightweight Directory Access Protokol
MBone	Multicast Backbone
MCU	Multipoint Control Units
MFC	Microsoft Foundation Classes
MIME	Multipurpose Internet Mail Extensions
MMusic	Multiparty Muiltimedia Session Control
Modem	Modulator-Demodulator
MPEG	Motion Picture Expert Group
ODBC	Opend Database Connectivity
OLE	Object Linking and Embedding
OLTP	Online Transaction Processing
ORB	Object Request Broker

PAL	Phase Alternating Line
PDF	Portable Document Format
RAID	Redundant Array of Inexpensive Disks
RMI	Remote Method Invocation
RTFM	Real-Time Traffic Flow Measurement Protocol
RTP	Real-Time-Protocol
SAP	Session Announcement Protocol
SDP	Session Deseription Protocol
SIP	Session Initiation Protokol
SMTP	Simpl Mail Transfer Protokol
SQL	Simply Query Language
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
URL	Uniform Resource Locator
WAN	Wide Area Network
WWW	World Wide Web

Literatur- und Quellenverzeichnis

[CSCW92-98] Conference Proceedings on Computer Supported Cooperative Work, Proceedings of CSCW 92, 94, 96, 98 http://www.acm.org/pubs/contents/proceedings/cscw/

[EDMEDIA] Online Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecom ED-MEDIA 96-98 http://ad.informatik.uni-freiburg.de/proceedings

[HTM92] T.Hoshi, Y.Takahashi, K.Mori : „An Integrated Multimedia Desktop Communikation an Collaboration Platform for Broadband ISDN“, Proceedings of Multimedia 92

[IET94] IETF. „Minutes of the multipart multimedia session control working group (mmusic)“, Proceedings of 29th IETF, 1994

[IRM99] Irmscher, Klaus : „Technische Infrastruktur und Nutzungsszenarien beim Einsatz von TeleTeaching“, in TeleTeaching: eine neue Komponente in der universitären Lehre, RAABE, 1999.

[JaT99] Technische Universität Dresden: „ Java Based Teleteaching Kit“, 1999. http://telet.inf.tu-dresden.de/JaTeK.htm.

[KERR98] Kerres: „ Multimediale und telemediale Lernumgebungen“, Oldenburg 1998.

[KLU96] Klute, Rainer: „Das World Wide Web“, Addison-Wesley 1996

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

[KYA95] Kyas, Othmar: „ ATM-Netzwerke“, DATACOM-Verlag 1995.

[LAB93] Bellcore Information Networking Research Laboratory. The Touring Machine System. : „Communications of the ACM“ 1993

[MR94] M. Mülhäuser,, T.Rüdebusch : „Context Embedding and Reuse in Cooperative-Software Development. In Jose L.Encarnacao, James D Foley, Ralf Guido Herrtwich, Hrsg.,cPerspektives of Multimedia Systems, 1994.

[MSDN00] Microsoft Developer Network Library 2000, Microsoft Corporation 2000.

[MVC98] „Microsoft Visual C++ 6.0, Programmierhandbuch“, Microsoft Press 1998.

[NGU98] G. Nguyen, Diplomarbeit, „Entwicklung eines Videotransportprotokolls über UDP“, Universität Leipzig 1998

[ORF99] Orfali, Robert; Harkey Dan; Edwards, Jeri: „Client/Server Survival Guide“,

Wiley Computer Publishing 1999.

[RENZ00] Renz, S.: „ Synchrone und asynchrone Kommunikationsdienste für das Distance-Learning-Tool OVID“, Diplomarbeit, Universität Leipzig, Institut für Informatik, Juli 2000.

[RFC760] Postel, J.: "Internet Protocol," RFC 760, 1980. http://www.cis.ohio-state.edu./htbin/rfc/rfc0760.html.

[RFC761] Postel, J., "Transmission Control Protocol," RFC 761, 1980. http://www.cis.ohio-state.edu./htbin/rfc/rfc0761.html.

[RFC768] Postel, J. : „User Datagram Protocol“, RFC 768, 1980. http://www.cis.ohio-state.edu./htbin/rfc/rfc0768.html.

[RFC791] Information Sciences Institute University of Southern California: „INTERNET PROTOCOL DARPA INTERNET PROGRAM PROTOCOL SPECIFICATION“ , RFC 791, 1981. http://www.cis.ohio-state.edu./htbin/rfc/rfc0791.html.

[RFC793] Information Sciences Institute University of Southern California: „TRANSMISSION CONTROL PROTOCOL DARPA INTERNET PROGRAM PROTOCOL SPECIFICATION“, RFC 793, 1981. http://www.cis.ohio-state.edu./htbin/rfc/rfc0793.html.

[RFC1122] Braden, R: „Requirements for Internet Hosts - Communication Layers“, RFC 1122, 1989. http://www.cis.ohio-state.edu./htbin/rfc/rfc1122.html.

[ROS93] Rose, Marshall T.: „Verwaltung von TCP/IP-Netzen“ , Prentice Hall 1993.

[SCHI97] Schilli, Michael: „Effektives Programmieren mit Perl 5“, Addison-Wesley 1997

[STR92] Stroustrup, Bjarne : „ Die C++ Programmiersprache“, Addison-Wesley 1992.

[STÜ94] H.J. Stüttgen: „Network evolution and multimedia communication“, Technical Report 43.9404, IBM European Networking Center Heidelberg, Heidelberg 1994

[STE98] Steinmetz: „Multimedia-Technologie: Einführung und Grundlagen“, Springer-Verlag 1998.

[QUI95] Quinn, Bob; Shute, Dave: „Windows Sockets Network Programming“, Addison-Wesley 1995.

[TAN 98] Tanenbaum, Andrew S.: „Computernetzwerke“, Prentice Hall 1998.

[TBR98] D.A. Tietze, A.Bapat: „Document-centric groupware for distributed governmental agencies“, Proceedings of the 10th Conference on Advanced Information Systems Engeneerin, Pisa 1998

[WSM91] K. Watabe, S.Sakata, K.Maeno : „Distributed Desktop Conferencing System with Multi-user Multimedia Interface“, IEEE JSAC, 9(4):531-539, 1991

[YOU97] Young, Michael : „Das Programmierbuch Visual C++ 5“, Sybex 1997

Danksagung

Diese Arbeit wurde am Lehrstuhl „Rechnernetze und Verteilte Systeme“ des Instituts für Informatik der Universität Leipzig angefertigt.

Herzlich danken möchte ich:

Herrn **Prof. Dr. K. Irmscher**

für die Vergabe der Aufgabenstellung, die wertvollen Diskussionen sowie für die konstruktive Kritik während der Fertigstellung der Arbeit,

Herrn **Dr. K. Hänßgen**

Für die Betreuung der Arbeit sowie für die schnelle Hilfe bei auftretenden Problemen.

Weiterhin bedanke ich mich bei meinen Kommilitonen Igor Kurkov und Stanislaw Renz für die wertvolle Tips während der Fertigstellung der vorliegenden Arbeit.

Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe.

Leipzig, _____ _____

Alexander Roskin