ETH Zürich
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Diplomarbeit

Implementierung eines verteilten Dateisystems für “CLIPPEE”

Kliment Stojanov
kliments@student.ethz.ch

Mai 2004

Professor: Prof. Roger Wattenhofer
Assistenten: Ruedi Arnold und
Keno Albrecht

Vorwort

Zusammenfassung

Die Diplomarbeit behandelt die Implementierung eines verteilten Dateisystems welches auf den bestehenden Client-Peer Prototypen `Clippee` aufbaut. Das verteilte Dateisystem bringt eine höhere Robustheit und Verfügbarkeit im Vergleich zur zentralen Form eines Dateisystems. Die vernetzten Peers von Clippee stellen für die angemeldeten Clients einen logischen Server dar und sorgen für Datenkonsistenz. Damit das Dateisystem unter dem Zielbetriebssystem Windows verwendet werden kann, wird in einem zweiten Schritt eine Anbindung über einen File Transfer Protocol (FTP) und einen Server Message Block (SMB) Server vorgenommen.

Der FTP Server funktioniert zuverlässig und schnell und erlaubt dem Benutzer über den Windows Explorer, oder andere FTP Clients, den Zugriff auf das Dateisystem. Das automatische `Refresh` des betrachteten Ordners und das Kopieren innerhalb des Servers werden vom FTP Protokoll nicht unterstützt. Mittels SMB wird das Dateisystem als neuer Laufwerksbuchstabe in den Windows Explorer eingebunden und von Windows wie eine lokale Festplatte behandelt. Der Zugriff auf das Dateisystem funktioniert grösstenteils, die noch realisierbaren Optimierungen beziehen sich auf die Bereiche Stabilität und Geschwindigkeit. Die Anbindung über einen Treiber wird an einem uns zur Verfügung gestellten Exempel von der Technischen Hochschule Rapperswil (HSR) betrachtet.

Danksagung

Ich danke meinen Betreuern Ruedi Arnold und Keno Albrecht für ihre wertvollen Informationen und Hinweise, welche sie zu meiner Arbeit beigesteuert haben. Für das Korrekturlesen der Dokumentation danke ich Tiziana Guatieri.

Inhaltsverzeichnis

Kapitel 1

Einleitung

1.1 Motivation

In den letzten Jahren hat der Einsatz verteilter Systeme stark zugenommen. Ein Grund dafür ist, dass aufgrund von einzelnen Rechnerausfällen das Arbeiten im Gesamtsystem in eingeschränktem Masse weiterhin möglich ist, während dies bei einem zentralen System zum Totalausfall führen kann. Eine Hauptaufgabe von verteilten Systemen ist die Speicherung von Daten, wodurch Benutzer gleichzeitig von verschiedenen Orten auf die Daten zugreifen können. Eine verbreitete Methode diese Funktion zu gewährleisten ist eine Schnittstelle zwischen einem verteilten System und dem Betriebssystem eines Computers bereit zu stellen. Beispiele für bestehende verteilte Dateisysteme sind: AFS [15], NFS [27], xFS [29], Ivy [2], Coda [16] und Farsite [1]. Die Semantik wird vom unterliegenden Betriebssystem abstrahiert und in das verteilte System integriert. Dadurch wird das verteilte System plattformunabhängig. In dieser Diplomarbeit geht es um die Implementierung eines verteilten Dateisystems für den Prototypen eines Peer-to-Peer basierten Systems namens `Clippee` [19].

Meine persönliche Motivation für diese Arbeit liegt vor allem im breiten Spektrum des Themengebietes begründet. Es umfasst durch die Einarbeitung in Clippee die Komponenten eines Peer-to-Peer Netzwerkes (Sockets, Locks, Nachrichten, ...) und durch die Erstellung eines verteilten Dateisystems die Funktionen und Hintergründe eines Dateisystems. Die Programme sind in Java implementiert worden und haben mir eine optimale Plattform zur Verbesserung und Festigung meiner Javakenntnisse geboten.

1.2 Aufgabenstellung

In einem ersten Schritt geht es darum, ein Dateisystem in Clippee zu integrieren. Clippee wurde in Java realisiert und unterstützt verteiltes Lesen und Schreiben von (kleinen) Objekten. Im jetzigen Zustand werden die Datenobjekte direkt durch ihren `Key` referenziert, aber sie referenzieren sich untereinander nicht. Es soll eine Datenstruktur generiert werden, die auf Ordner und Dateien bezogen eine passende Semantik anbietet.

Um das Dateisystem im Zielbetriebssystem Windows XP verwenden zu können, soll eine Schnittstelle implementiert werden, die das Clippee Dateisystem in Windows integriert. Im Optimalfall wird Clippee im Windows Explorer als neuer Laufwerksbuchstabe (z.B. als `Clippee X:`) abgebildet.

1.3 Überblick

Im nächsten Kapitel werden einige Technologien zur Anbindung an das Betriebssystem erklärt und bewertet. Dadurch erhält man ein Bild von möglichen Lösungen, deren Vor- und Nachteile aufgezeigt werden. In Kapitel 3 wird die Implementierung des verteilten Dateisystems und das darunterliegende Clippee Netzwerk vorgestellt. Die Anbindung des Dateisystems an Windows durch FTP wird in Kapitel 4 und die Anbindung durch SMB in Kapitel 5 erklärt. In der Zusammenfassung werden die Schlussfolgerungen dieser Arbeit gezogen und anschliessend werden die Verbesserungs- und Erweiterungsvorschläge der Implementierungen im Ausblick erläutert. Im Anhang befindet sich eine Anleitung für das Installieren und Starten der Programme. Weiter sind detailliertere Informationen zu den Implementierungen aufgeführt.

Kapitel 2

Grundlagen und Technologieüberblick

Ziel ist, ein fremdes Dateisystem in das lokale Betriebssystem einzubinden, in unserem Fall ist das die Integration des Clippee Dateisystems in das Windows Betriebssystem. Aufgrund der zunehmenden Verbreitung des Internets und der damit verbundenen Computervernetzung im Allgemeinen, stellt die oben erwähnte Integration kein neues Problem dar. Im folgenden werden einige bestehende Lösungen aufgeführt und analysiert: Wo liegen die Vor- und Nachteile? Sind die Lösungen umsetzbar?

2.1 Server Message Block (SMB)

2.1.1 Geschichte

Das Server Message Block (SMB) Protokoll [24] wurde 1980 von Dr. Barry Feigenbaum von IMB Corporation konzipiert [13]. Später wurde es von verschiedenen Firmen wie 3Com, IBM, Intel und Microsoft erweitert. Mitte der 90er Jahren wurde das Protokoll zu Common Internet File System (CIFS) umbenannt. Heutzutage werden beide Namen, SMB und CIFS, für das Protokoll verwendet. Das Protokoll dient den Client-Anwendungen eines Computers um Dateien auf einem anderen Computer des Netzwerkes lesen und speichern zu können. Mittels SMB Protokoll ist auch der Zugriff auf weitere Ressourcen des Servers, wie z.B. Drucker, Mailslots und benannte Pipes, gewährleistet.

Microsoft Windows Betriebssysteme beinhalten seit Windows 95 einen SMB Client und einen SMB Server. Durch die Integration von SMB in das Windows Betriebssystem hat eine starke Verbreitung stattgefunden. Das Einsatzgebiet des Protokolls wird durch das Open Source Projekt `Samba` [32] zusätzlich erweitert und bietet dadurch sowohl einen Server als auch einen Client für Linux Systeme. Dies ermöglicht den Zugriff von Windows auf das Dateisystem von Linux und umgekehrt.

Das Protokoll hat sich im Laufe der Jahre stets weiterentwickelt, und entsprechend bestehen verschiedene Versionen. So sprechen z.B. Microsoft Windows for Workgroups 3.1 und Windows NT nicht die gleiche Protokollvariante. Die aktuelle Protokollvariante wird in ihrer heutigen Form meistens unter dem Namen CIFS referenziert. Die Weiterentwicklung seitens Microsoft stellt ein Problem dar, da die neuen Dokumentationen aufgrund von Lizenzierungsrestriktionen oft ungenügende Informationen enthalten [28]. Dadurch wird die Entwicklung von SMB Produkten anderer Parteien als Microsoft erheblich erschwert.

2.1.2 Windows und SMB

Die meisten Leser kennen wahrscheinlich die Option `Map Network` (siehe Abb. 2.1) des Windows Explorers. Mit dieser wird ein freigegebener Ordner eines anderen Computers als neues Laufwerk mit Laufwerksbuchstaben abgebildet. Das ermöglicht das Browsen und die Dateienbearbeitung, als würde es sich um ein lokales Laufwerk handeln. Die Option `Map Network` wiederum verwendet den DOS-Befehl `net use`:

C:\ net use [Laufwerksbuchstabe:] {\ \ Server \ Verzeichnis}

Dieser basiert auf dem SMB Protokoll und ermöglicht den Dateizugriff zwischen zwei Computern.

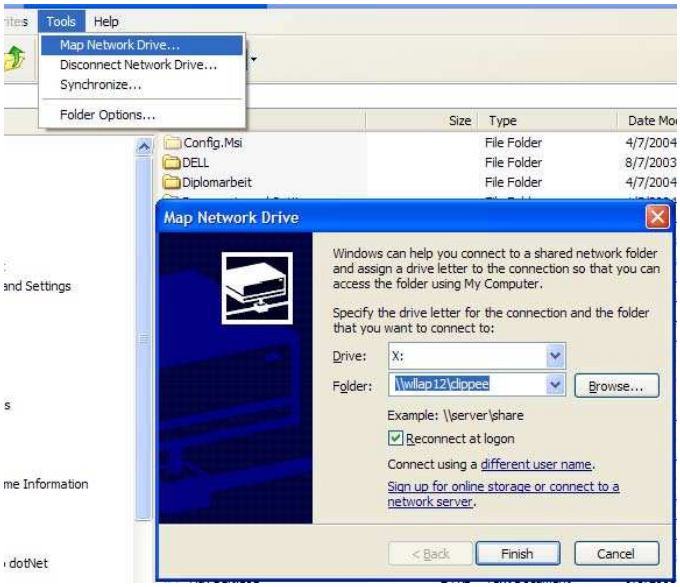

Abbildung 2.1: Mit dem Befehl `Map Network` lassen sich freigegebene Ordner anderer Computer im Windows Explorer abbilden.

2.1.3 Transport

Der Befehl `net use` verwendet für den Transport NetBIOS über TCP/IP (oft als NBT oder NetBT bezeichnet). Für den Verbindungsaufbau zwischen Client und Server werden die NetBIOS Namen[1] verwendet. Wenn die Verbindung erstellt ist, kann der Client mittels SMB Befehlen auf freigegebene Ressourcen zugreifen, Dateien öffnen, Dateien lesen und schreiben. SMB kann ausser über NBT auch über andere Transportprotokolle abgewickelt werden, wie z.B. IPX, NetBEUI und DECnet, und auch direkt über TCP/IP [28].

Aufbau der NetBIOS Pakete

Der Namensservice von NetBIOS erlangt aufgrund des NetBIOS Namens die dazugehörige IP Adresse des Servers. Danach wird eine NetBIOS Session erstellt, über welche die SMB Pakete verschickt werden. NetBIOS Nachrichten bestehen aus einem Header (siehe

[1] NetBIOS Namen dienen zur Identifizierung von Computern und dürfen maximal 15 Buchstaben lang sein.

Abb. 2.2) und einem Datenblock. Das Feld `Typ` (1 Byte) bezeichnet die Art der Nachricht (Session Request, Session Message ...), das Feld `Length` bestimmt die Grösse des Datenblocks.

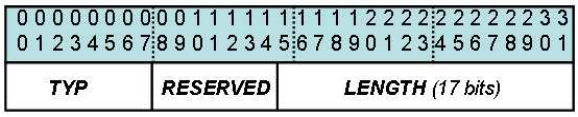

Abbildung 2.2: Der Header der NetBIOS Pakete besteht aus vier Bytes.

Aufbau der SMB Pakete

Die SMB Nachrichten werden im Datenblock der NetBIOS Nachrichten mitverschickt, und in drei Hauptblöcke aufgeteilt:

- Header
- Parameterblock
- Datenblock

Der SMB Header (siehe Abb. 2.3) besitzt immer die fixe Grösse von 32 Bytes und beginnt immer mit dem Byte `0xFF` gefolgt von der ASCII Zeichenkette "SMB".

COMMAND: Dieses Byte repräsentiert den SMB Befehl, die Art des Pakets. Beispiele: `Smb_Com_Delete`(Byte:0x06), `Smb_Com_Read_AndX`(Byte:0x2E) oder `Smb_Com_Negotiate`(Byte:0x72).

STATUS: Dieses Feld wird für die Error Codes verwendet.

FLAGS und FLAGS2: Definieren spezielle Attribute, bspw. ob das Paket eine Anfrage oder Antwort ist, Gross- und Kleinschreibung unterschieden wird usw.

TID: Tree ID wird vom Server zugewiesen, nachdem der Client eine erfolgreiche Verbindung zum freigegebenen Ordner hergestellt hat.

PID: Process ID identifiziert den anfragenden Prozess des Clients.

UID: User ID wird vom Server nach einer erfolgreichen Session Setup Anfrage mit gültigem Benutzername und Passwort zugewiesen. UID gilt für die Gesamtdauer einer Session.

MID: Multiplex ID kennzeichnet die unterschiedlichen Anfragen eines Prozesses. Somit ist die Zugehörigkeit von Anfragen und Antworten jederzeit transparent.

Der Parameterblock (siehe Abb. 2.4) wird für befehlsspezifische Parameter verwendet. Da nicht jeder Befehl über die gleiche Anzahl und Art von Parametern verfügt, ist die Blocklänge variabel und wird durch das Feld `WordCount`[2] definiert.

Der Datenblock (siehe Abb. 2.5) gleicht dem Parameter Block. Auch hier ist die Länge variabel und durch das Feld `ByteCount` definiert. Der Hauptunterschied ist, dass der Parameter Block nur einen kleinen Teil von Daten enthält (Parameter), während der Datenblock normalerweise bedeutend grössere Mengen an Daten enthält (z.B. Dateiinhalte).

[2] Ein `Word` entspricht zwei Bytes.

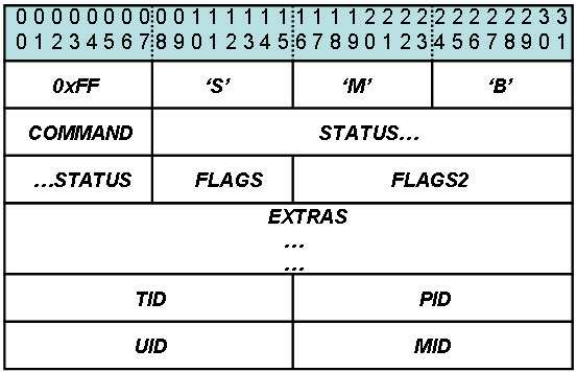

Abbildung 2.3: Der SMB Header besitzt eine feste Grösse von 32 Bytes und beginnt immer mit dem Byte `0xFF` und der ASCII Zeichenkette “SMB”.

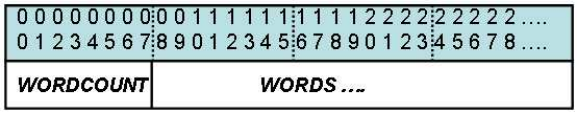

Abbildung 2.4: Der SMB Parameterblock enthält die befehlsspezifischen Parameter.

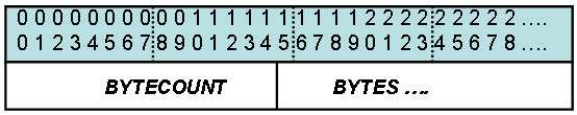

Abbildung 2.5: Der SMB Datenblock kann grosse Mengen an Daten enthalten (z.B. Dateiinhalt).

2.1.4 Lösungsidee

Eine mögliche Lösung für die Anbindung ist die Implementierung eines SMB Servers mit Verbindung zu Clippee. Dadurch kann das Clippee Dateisystem auf einen neuen Laufwerksbuchstaben unter dem Windows Explorer abgebildet werden. Der Clippee SMB Server empfängt die Anfragepakete des Windows SMB Clients und kreiert die Antwortpakete aufgrund der von Clippee gelieferten Daten.

2.1.5 Vorteile und Nachteile der SMB Lösung

Vorteile

- Durch die Integration von SMB Server und Client in Windows Betriebssysteme und Samba unter Linux Computern, ist das Protokoll sehr verbreitet.
- Das Clippee Dateisystem bekommt einen eigenen Laufwerksbuchstaben und wird wie eine lokale Partition behandelt.
- Automatisches `Refresh`: Veränderungen werden beim Client sofort angezeigt.

Nachteile

- Damit man die ankommenden SMB Anfragen behandeln kann, muss beim Peer mit dem laufenden Clippee SMB Server der NetBIOS Dienst abgeschaltet werden.
- Teilweise Unübersichtlichkeit des Protokolls: SMB besteht unterdessen aus mehr als 100 Befehlen mit doppelt belegten Funktionen und undokumentierten Parametern.

2.2 File Transfer Protocol (FTP)

2.2.1 Geschichte

Das File Transfer Protocol (FTP) ist ein sehr verbreitetes Protokoll in der Computerwelt und wurde im Jahre 1971 am Massachusetts Institute of Technology (MIT) entwickelt. FTP ist auch ein Teil der `Internet Protocol Suite`[3] und ermöglicht den Dateiaustausch zwischen Computern mit verschiedenen Betriebssystemen. FTP ist im Request For Comments (RFC) 959 [17] der Internet Engineering Task Force (IETF) [9] spezifiziert.

Die meisten Webbrowser beherrschen heutzutage das FTP Protokoll und ermöglichen damit Zugriffe auf FTP Server innerhalb des Browsers. Es gibt unzählige kommerzielle und frei verfügbare FTP Clients. Auch der Windows Explorer ermöglicht den Zugang zu einem FTP Server.

2.2.2 Funktionsweise

Das FTP Protokoll ist nicht sehr kompliziert gestaltet und daher leicht verständlich. Sobald der Client den Server kontaktiert, schickt dieser eine Willkommensmeldung. Danach meldet sich der Client entweder mit einem gültigen Benutzernamen und Passwort oder als `Anonymous` und Email-Adresse, beim Server an. Als anonymer Benutzer kann der Client normalerweise nur auf den öffentlichen Ordner zugreifen und nichts auf dem Server speichern.

Der Client schickt über die Klartext-Verbindung die Anfragen in Form von simplen ASCII Zeichenketten, gefolgt von allfällig benötigten Parametern. Der Server seinerseits antwortet mit dreistelligen Zahlencodes, welche bekanntgeben, ob die Operation erfolgreich verlaufen oder ein Fehler aufgetreten ist. Die Meldungen nach den Zahlencodes werden im Normalfall nicht vom Client Programm verwendet, sie dienen einzig der Verständlichkeit für den Benutzer. In der Tabelle 2.1 sieht man zur Veranschaulichung einen Beispieldialog der Anmeldungsprozedur zwischen einem Server und einem Client.

Server:	220 Sample FTP server ready. Please give user-name
Client:	USER Anonymous
Server:	331 User name OK. Please give your email address as password.
Client:	PASS joe@nowhere.com
Server:	230 User logged in

Tabelle 2.1: Dialog zwischen Server und Client während der Anmeldeprozedur.

Das FTP Protokoll ist so konzipiert, dass grundsätzlich zwei Verbindungen zwischen dem Server und dem Client hergestellt werden: eine für Befehle und Antwortcodes, die andere

[3] Satz von Protokollen welche den Protokoll Stack implementieren, auf welchem das Internet läuft. Es taucht oft unter dem Namen `TCP/IP protocol suite` auf, nach den beiden wichtigsten und ersten definierten Protokolle der Suite benannt.

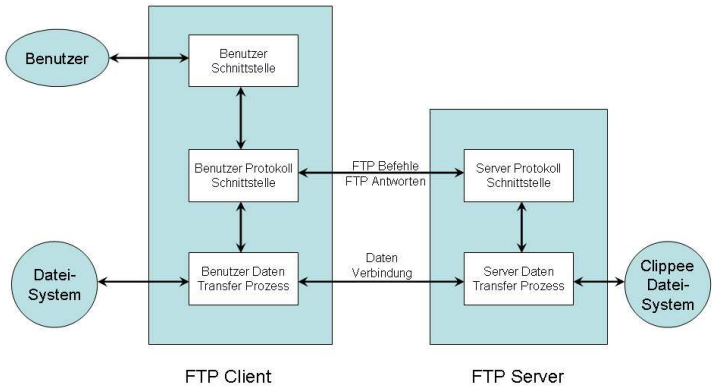

Abbildung 2.6: FTP Model: Die Befehle und Antwortcodes werden über die Kommando Verbindung übertragen, während die Daten über eine zweite Verbindung verschickt werden.

für die Übertragung der Daten (siehe Abb. 2.6). Die Verbindung mit den Befehlen läuft standardmässig über den Port 21, während die Datenübertragung über einen zufälligen, vom Server bestimmten Port abgewickelt wird. Die Herstellung einer Datenverbindung zwischen einem Client und einem Server ist in der Tabelle 2.2 abgebildet.

Client:	TYPE A
Server:	200 Type set to A
Client:	PASV
Server:	227 Entering passive mode (193,91,161,12,28,46)
Client:	LIST
Server:	150 Opening ASCII mode data connection for /bin/ls
Server:	226 Transfer complete

TYPE A	*Client:* Sagt dem Server die Verzeichnisauflistung im ASCII Format zu senden
PASV:	*Client:* Der Server soll eine neue Datenverbindung erstellen und sich auf eine Datenübertragung vorbereiten. *Server:* Gibt die IP Adresse (193.91.161.12) und die Port Nummer ($28 * 256 + 46 = 7214$) der Datenverbindung an.
LIST:	*Client:* Der Server soll die Verzeichnisauflistung übertragen.

Tabelle 2.2: FTP Dialog zwischen Client und Server für die Herstellung einer Datenverbindung.

2.2.3 Vorteile und Nachteile der FTP Lösung

Vorteile

- Es sind für praktisch alle Betriebssysteme FTP Clients verfügbar.
- Einfaches Protokoll und daher leicht zu implementieren.
- Kann in Windows Explorer unter `Netzwerke` eingebunden werden.
- Kann innerhalb der Webbrowser verwendet werden.

Nachteile

- Es ist kein Kopieren/Verschieben innerhalb des Servers möglich, nur zwischen Client und Server.
- Kein automatisches `Refresh`. Der Inhalt des Ordners wird erst aktualisiert wenn der Client eine Anfrage schickt.
- Dateien und Passwörter werden grundsätzlich im Klartext geschickt (Ausnahme ist SFTP: Erweiterung von FTP mittels SSH).

2.3 Treiber (Windows IFS Kit)

2.3.1 Allgemein

Bei der Suche nach passenden Lösungen zur Anbindung des Clippee Dateisystems an Windows bin ich auf Programme gestossen, welche andere Dateisysteme ins Windows Betriebssystem einbinden. Es gibt Programme die anhand von ISO Dateien virtuelle CD-Roms simulieren, andere Programme binden betriebsfremde Dateisysteme ein.

Die Funktionsweise ist dabei die gleiche: Es wird ein Treiber verwendet welcher das fremde Dateisystem in das Betriebssystem integriert. Dadurch können Dateisysteme wie ISO9660 [8] (CD Rom Dateisystem), Ext2 [26](Linux Dateisystem), Ext3 [34](erweitertes Linux Dateisystem), NTFS [22](Windows Dateisystem) oder andere Dateisysteme eingebunden werden, meistens als separates Laufwerk. Zum Beispiel: Mount Everything und CD-Rom Emulator von Paragon [12], DAEMON Tools [33] und Ext2Fsd [21]. Für die Programmierung von Dateisystemtreibern für Windows benötigt man das Windows Installable File System (IFS) Kit, welches uns freundlicherweise von Microsoft für diese Arbeit zur Verfügung gestellt wurde.

2.3.2 Vorteile und Nachteile der Treiber Lösung

Vorteile

- Direkte Einbindung ins Windows Betriebssystem.
- Kann eigenen Laufwerksbuchstaben für Clippee Dateisystem verwenden, und alle Funktionen werden unterstützt (`Refresh`, Kopieren/Verschieben,..).

Nachteile

- Lösung ist unbrauchbar für andere Betriebssysteme als Microsoft Windows.

Kapitel 3

Das Clippee Dateisystem

Das Dateisystem baut auf `Clippee` auf. Für ein besseres Verständnis und um die auftretenden Probleme des Dateisystems besser verstehen zu können, werden die Clippee Komponenten vorgestellt. Danach werden Aufbau und Funktionsweise des Dateisystems erklärt.

3.1 Clippee

3.1.1 Topologie Komponente

Der Client Peer Prototyp Clippee besteht aus einer Menge von Peers, die als kompletter Graph verbunden sind (siehe Abb. 3.1). Die Menge der Peers dient als logischer Server für alle angemeldeten Clients. Durch den kompletten Graph der `n` Peers ist die Nachrichtenübertragung bis und mit einem Ausfall von n-2 Verbindungen gewährleistet. Falls die direkte Verbindung zwischen zwei Peers ausfällt, wird eine Zwei-Hop-Verbindung via einem anderem Peer gesucht.

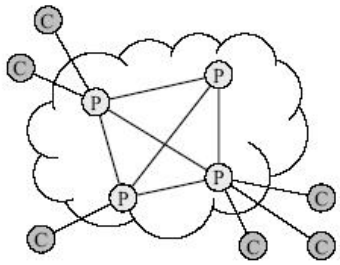

Abbildung 3.1: Die Peers bilden als vollständiger Graph das Clippee Peer-to-Peer Netzwerk, und stellen den Clients verschiedene Dienste zur Verfügung.

3.1.2 Netzwerk Komponente

Die Nachrichten zwischen den Peers werden über TCP-Verbindungen ausgetauscht, entweder direkt oder über die oben erwähnte Zwei-Hop-Verbindung. Die ankommenden Nachrichten werden von einer vorgegebenen, fixen Anzahl von Threads behandelt. Falls es kei-

nen freien Thread gibt, wird die Nachricht ignoriert. Der Sender verfügt über ein festgelegtes Timeout für Nachrichten, welche eine Antwort benötigen. Nach Ablauf des Timeouts wird die Nachricht vom Sender neu verschickt oder eine Fehlerbehandlung gestartet. Das Ausbleiben der Antwort kann Ursache von einem Absturzes, dem Löschen der Nachricht oder einer Verzögerung sein.

3.1.3 Daten Komponente

Clippees Datenobjekte bestehen aus einem Key, der die Objekte eindeutig identifiziert, einer Versionsnummer, die bei jeder Veränderung inkrementiert wird, und einem Bytearray, welches die Daten beinhaltet. Für die Speicherung der Datenobjekte wird ausschliesslich der Arbeitsspeicher benutzt, die Daten befinden sich zu keiner Zeit auf der Festplatte. Wenn sich alle Peers vom Netzwerk abmelden sind alle Daten undwiderruflich verloren. Die Daten werden vollständig repliziert, das heisst dass jeder Peer jedes Datenobjekt speichert. Zur Replikation wird ein `eager replication` Algorithmus verwendet; es werden alle Kopien des Datenojbektes anhand der ursprünglichen Transaktion aktualisiert. Dadurch erhöht sich die benötigte Zeit zum Speichern des Objektes linear mit dessen Grösse. Die Leseoperation basiert auf einem lokalen Lookup des Peers.

Für die Schreiboperation muss der Peer einen Lock (ein Token) für dieses Objekt erlangen, indem er an alle Peers eine `AcquireLock`-Nachricht verschickt. Er erlangt den Lock nur, wenn alle Peers ihm den Lock gewähren oder nicht innerhalb der vorgegebenen Zeit antworten. Dieser optimistische Ansatz garantiert keinen eindeutigen Lock, funktioniert in der Praxis aber sehr zufriedenstellend. Wenn der Lock erlangt wurde, kann ein Peer das Datenobjekt verändern bis er den Lock wieder freigibt. Um die mögliche Inkonsistenz der Daten zu beheben, läuft im Hintergrund ein `Lazy Replication`-Prozess, der zufällige Objekte von verschiedenen Peers vergleicht und gegebenfalls abgleicht.

3.2 Implementierung des Clippee Dateisystems

Das Clippee Dateisystem baut auf den bestehenden Clippee Komponenten auf, und erweitert diese passend. Das `DataObject` wird zu einem `FilesystemObject` und `DirectoryObject` erweitert. Die Schnittstelle zum Dateisystem bildet der `FilesystemManager`, welcher eine Erweiterung des `DataManagers` ist (siehe Abb. 3.2). Zur Kommunikation werden die Nachrichten der erweiterten Netzwerk Komponente verwendet. Im Hinblick auf die Anbindung an den Windows Explorer wird aus Zeitgründen ein Dateisystem mit den Grundfunktionen implementiert. Das Dateisystem lässt sich mit Zusatzfunktionen wie Zugriffsrechte, erweiterte Fileattribute usw. zukünftig erweitern.

3.2.1 Filesystem Object und Directory Object

Diese Objekte stellen die eigentlichen Dateien und Ordner dar, mit denen der Benutzer letztendlich im Windows Explorer arbeitet.

Das `FilesystemObject` (FSObject) ist eine direkte Ereweiterung des `DataObjects` von Clippee, und das `DirectoryObject` (DirObject) eine Erweiterung des `FSObjects`. Für einige Funktionen des Dateisystems ist die Unterscheidung zwischen Datei (`FSObject`) und Ordner (`DirObject`) nötig, denn die `DirObjects` enthalten die Keys der Unterordner und der Dateien. Dateien und leere Ordner repräsentieren die Blätter der Baumstruktur des Dateisystems. Der `Rootordner` stellt die Wurzel des Baumes dar und wird vom ersten Peer im Clippee Netzwerk erstellt. Danach kann er nicht mehr gelöscht

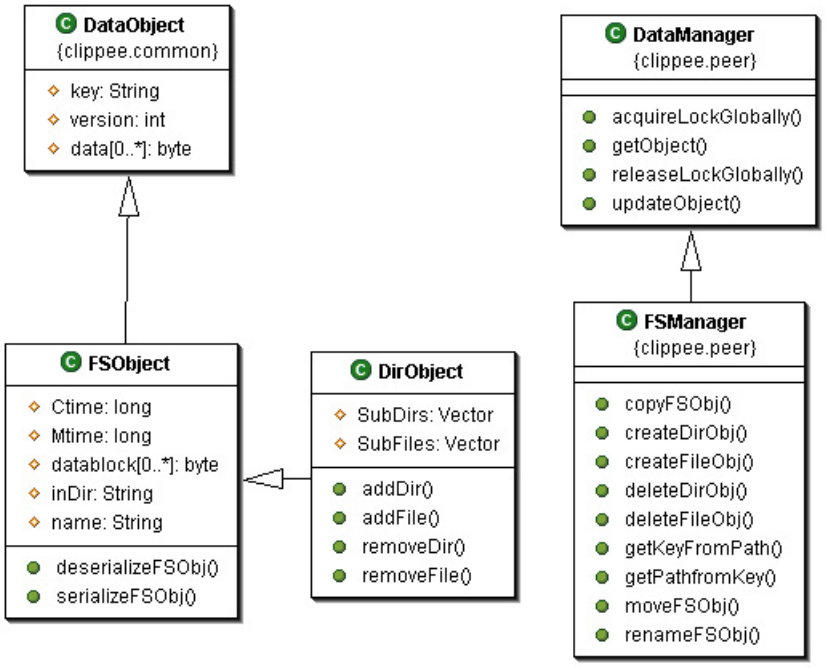

Abbildung 3.2: UML Schema mit den wichtigsten Attributen und Funktionen der Clippee Dateikomponenten.

werden. Da der `Rootordner` als Einstiegspunkt in die Dateistruktur dient, bekommt er einen vordefinierten Key.

Jedes andere Objekt im Dateisystem erhält bei der Erstellung einen eindeutigen Key zur Identifizierung, welcher im `DataObject` gespeichert wird. Der Key wird aus der Systemzeit und der IP Adresse des Peers zusammengestellt. So wird sichergestellt dass kein anderer Peer ein Objekt mit dem gleichen Key erstellt. Eine Hashtabelle im `FSManager` dient zur Speicherung der Objekte. Mittels des Keys erhält man die gewünschten Objekte aus der Hashtabelle. Gleichzeitig dient er als Referenz innerhalb der Baumstruktur. Durch die Abstrahierung des absoluten Pfades von der Baumstruktur werden beim Umhängen eines Teilbaumes (Verschiebungsoperation) nur die direkt betroffenen Knoten verändert. Für die Knoten des umgehängten Teilbaumes verändert sich nichts, denn der sich geänderte absolute Pfad wird direkt aus der momentanen Baumstruktur ermittelt. Die `FSObjects` werden innerhalb der Peers nur als serialisierte Bytestreams verschickt. Auch die Speicherung in der Hashtabelle erfolgt nur als `DataObject`, welches alle Informationen des `FSObjects` und `DirObjects` in serialisierter Form im Byte Array `Data` gespeichert hat.

Das `FSObject` besitzt die üblichen Fileattribute, wie z.B. Name, Datum der Erstellung, Grösse und Inhalt der Datei oder des Ordners. Das Attribut `inDir` enthält den Key des übergeordneten Ordners, d.h. des Vaterknotens. Durch iteratives aneinanderreihen der Namen bis zum `Rootordner` erhält man so aus dem Key den absoluten Pfad. Durch die serialisierte Speicherung der `FSObjects` lassen sich zusätzliche Attribute leicht hinzufügen. Mögliche Attribute sind: Gruppen ID, Benutzer ID, lesbar, schreibbar, sichtbar und Andere.

Das vom `FSObject` erweiterte `DirObject` ist mittels den zwei zusätzlichen Attributen `SubDirs` und `SubFiles` für die Baumstruktur verantwortlich. In diesen werden die Keys der Unterordner und der Dateien gespeichert, welche in der Baumstruktur die Teilbäume

und Blätter repräsentieren. Die Grösse des `DirObject` ist immer gleich Null, da es als Ordner keinen Dateninhalt besitzt.

3.2.2 Filesystem Manager

Der Filesystem Manager (`FS Manager`) stellt die Schnittstelle zwischen dem Clippee Dateisystem und den Prozessen, die darauf zugreifen möchten, dar. Der `FS Manager` ist eine Erweiterung des `DataManagers`, welcher das gesamte Dateisystem in Form von `DataObjects` und deren Keys in der Hashtabelle speichert. Weitere Funktionen des `DataManagers` sind Updates der `DataObjects`, das Erlangen und Freigeben von Locks und das Löschen von `DataObjects` innerhalb von Clippee.

Einige Basisfunktionen des FS Managers

Bei einigen Methoden können aufgrund der Locks und der Parameter verschiedene Fälle auftreten. Um welche Fälle es sich dabei handelt und wie sie gehandhabt werden, wird in der Tabelle 3.1 erklärt.

Funktion	*Parameter*	*Fehlerfall*
Erstellen	Zielordner Key, Name des neuen Objektes	Falls Objekt mit diesem Namen im Zielordner bereits existiert, wird das Objekt nicht erstellt.
Löschen	Zielordner Key, Name des zu löschenden Objektes	Falls der Lock für den Zielordner nicht erlangt werden kann, wird das Objekt nicht gelöscht. Falls Locks innerhalb eines zu löschenden Ordners nicht erlangt werden können, werden diese Objekte mit den übergeordneten Ordnern nicht gelöscht.
Verändern	Key, Attributwerte	Wenn der Lock für das Objekt nicht erlangt werden kann, wird die Veränderung nicht durchgeführt.
Verschieben	Key des Objektes, Zielordner Key	Falls der Lock für das zu verschiebende Objekt, dem alten Ordner oder dem neuen Zielorder nicht erlangt werden kann, wird das Objekt nicht verschoben. Es wird absichtlich darauf verzichtet die Locks aller involvierten Objekte zu erlangen.[1] Die Konsistenz wird durch die Verschiebung eines gelockten Objektes nicht verletzt. Die Struktur unter den Objekte bleibt die gleiche, bloss die absoluten Pfade ändern sich.[2]
Kopieren	Key des Objektes, Zielordner Key	Misslingt, falls der Lock des Zielordners nicht erlangt werden kann.[3]

Tabelle 3.1: Grundfunktionen des FS Managers und auftretende Fehlerfälle.

[1] Dies würde dazu führen, dass es aufgrund anderer Benutzer sehr schwierig wäre einen Ordner mit vielen Dateien und Unterordnern zu verschieben.

[2] Der Inhaber des Locks muss das Objekt nach der Verschiebung jedoch neu “lokalisieren”.

[3] Involvierte Objekte werden nicht geändert, nur der Zielordner.

3.2.3 Clippee Shell

Damit ein Benutzer das Clippee Dateisystem benutzen oder testen kann, wurde eine Shell im Stile eines DOS Eingabefensters implementiert (siehe Abb. 3.3). Die Shell schickt die Anfragen des Benutzers an den FS Manager und stellt die Antworten für den Benutzer in verständlicher Form dar. Damit können die Funktionen des Dateisystems getestet und allfällige Fehler frühzeitig erkannt werden.

Die Shell deckt alle Funktionen des FS Managers ab und stellt damit eine erste Schnittstelle zwischen dem Benutzer und dem Clippee Dateisystem dar. Natürlich ist die Handhabung durch die Textbefehle nicht sehr komfortabel, es handelt sich nur um eine lokale, auf den Peers laufende Lösung.

Mit der Shell lassen sich bereits ganze Dateistrukturen via Clippee zwischen Computern kopieren. Durch die Shell lassen sich Ordner von der lokalen Festplatte ins Clippee Dateisystem kopieren. Das Clippee System verteilt die Daten über das Netzwerk an alle Peers. Danach kann auf einem anderen Peer via Shell das komplette Clippee Dateisystem oder einen Teil davon auf dessen lokale Festplatte abgespeichert werden.

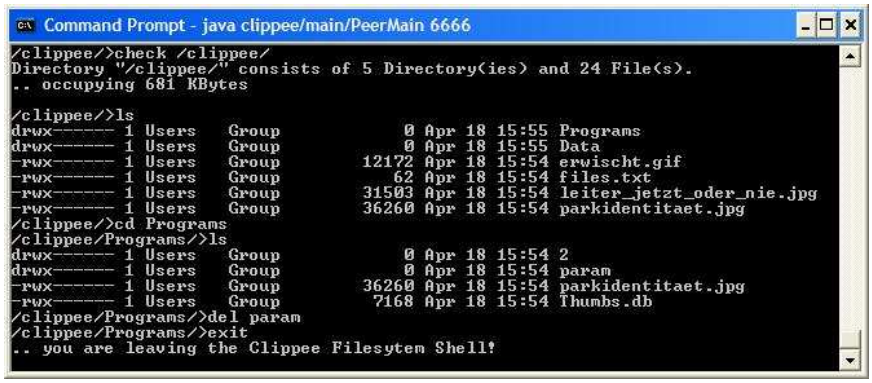

Abbildung 3.3: Screenshot der Clippee Shell.

Kapitel 4

Anbindung durch FTP

In den nächsten zwei Kapiteln werden die Lösungen der zweiten Hauptaufgabe dieser Diplomarbeit behandelt: die Anbindung an das Betriebssystem Microsoft Windows. Das Clippee Dateisystem soll unter Windows benutzbar sein und auf möglichst vielen anderen Betriebssystemen unterstützt werden.

Aufgrund aufgetauchter Probleme bei der Implementierung der SMB Lösung und speziell benötigter Software für die Treiber Lösung, wurde als erstes die FTP Lösung implementiert.

4.1 Lösungsansatz

Wir versuchen einen FTP Server zu implementieren, welcher unter Clippee läuft und die ankommenden FTP Anfragen der Clients bearbeitet. Da Clippee in Java implementiert ist, soll der Server für eine nahtlose Anbindung auch in Java implementiert sein. Der Server stellt eine separate Komponente des Clippee Systems dar und soll beim Start eines Peers optional gestartet werden können.

4.2 Implementierung

Beim Versuch einen FTP Server von Grund auf neu zu implementieren hat sich gezeigt, dass dieses Unterfangen trotz Einfachheit des Protokolls ziemlich zeitaufwändig ist. Deshalb haben wir uns entschieden nach einem Open Source FTP Server in Java zu suchen und für unsere Bedürfnisse anzupassen.

Durch die Open Source Bedingung ist die Menge der möglichen Server stark eingeschränkt worden. Dabei hat sich der FTP Server von `JQ-DATA` [18] für unsere Zwecke am geeignetsten erwiesen.

Der FTP Server erfüllt nicht nur unsere Anforderungen sondern bringt auch eine grafische Benutzeroberfläche mit vielen zusätzlichen Einstellungsmöglichkeiten mit sich (siehe Abb. 4.1). In erster Linie interessieren wir uns für die Grundfunktionen (Browsen, Dateitransfer, ...). Es ist gut vorstellbar einige der Zusatzfunktionen (Benutzerunterscheidung, ...) in einem späteren Zeitpunkt in Clippee zu integrieren.

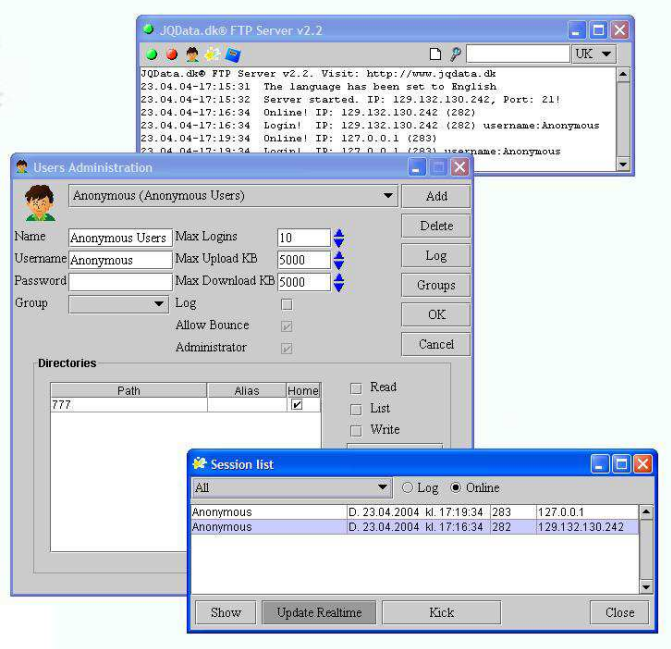

Abbildung 4.1: Ein Screenshot der Benutzeroberfläche des FTP Servers.

Einige Features des FTP Servers

- Regulierung der Up- und Downloadgeschwindigkeit auf Benutzerebene.
- Blockieren einer oder mehrerer IP Adressen.
- Mögliche Einteilung der Benutzer in Benutzergruppen.
- Limitierung der Anzahl gleichzeitiger Benutzer auf dem Server.

Systemanforderungen

- Java Development Kit (JDK) 1.4
- Windows 9x / Windows 2000 / Windows XP (auf anderen Plattformen noch nicht getestet)

4.2.1 Anpassung der Einstellungen und der Methoden

Das Clippee Dateisystem unterscheidet noch keine Benutzer und dazugehörende Passwörter. Da wir die Benutzerverwaltung nicht dem FTP Server überlassen möchten, erstellen wir ein anonymes Benutzerkonto auf dem Server und als `Rootordner` wird das Verzeichnis `/clippee/` angegeben. Nun können alle Benutzer ohne Passwort auf das Clippee Dateisystem zugreifen. Weiter werden die Anzahl der gleichzeitigen Benutzer und die erlaubte Datenbandbreite heraufgesetzt.

Die Peers können den FTP Server optional durch den Aufruf der Klasse `Run` (siehe Abb. 4.2) starten. Die benötigten Daten des Clippee Dateisystems werden über den FS Manager von Clippee erlangt. Die Klasse **`Server`** wird als Singleton implementiert und verwaltet die Benutzer und die Verbindungen über die Klasse `Dock`.

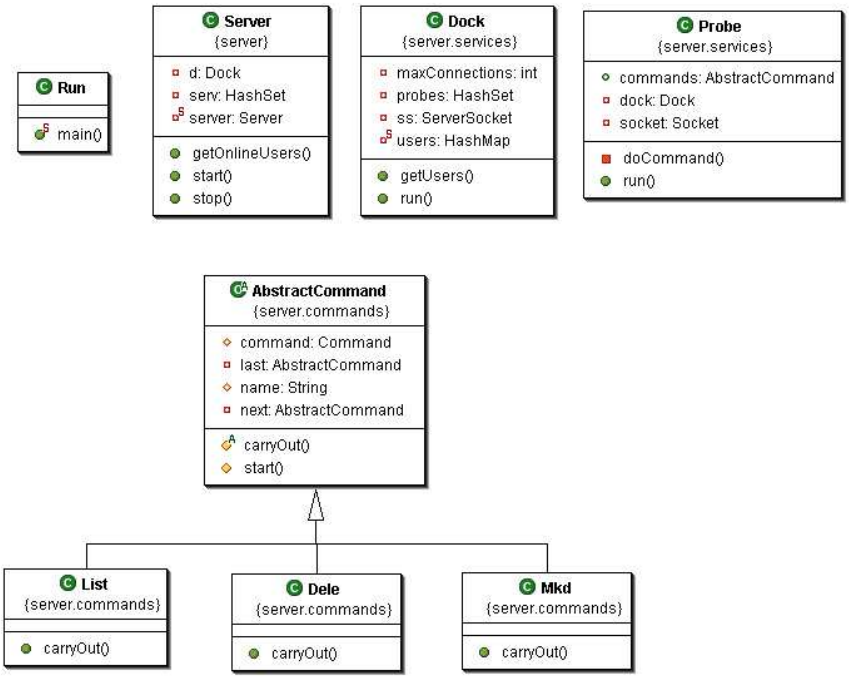

Abbildung 4.2: UML Schema der wichtigsten Klassen und Methoden des FTP Servers. Die FTP Befehle erweitern die Klasse `Abstract Command` und werden in einer Liste verwaltet.

`Dock` horcht auf dem Port (standardmässig 21) und stellt die Verbindungen zu den Clients über die Klasse `Probe` dar. Er verwaltet auch die angemeldeten Benutzer und stellt sicher, dass nur die vorgegebene Anzahl von Benutzern gleichzeitig auf den Server zugreift.

In **`Probe`** wird die Anfrage des Clients empfangen und behandelt. Alle implementierten Befehle werden durch eine Klasse repräsentiert und sind in einer Auflistung gespeichert. Bei einer Anfrage wird die Auflistung der Befehle durchgegangen und die dazugehörende Klasse ausgeführt, wobei die Antwort anhand der Daten aus dem Clippee Dateisystem erstellt und über den Socket an den Client zurückgeschickt wird. Wenn der Befehl nicht implementiert ist (nicht in der Auflistung), wird eine Fehlermeldung an den Client geschickt. Falls der Client keine weiteren Anfragen stellt oder das festgelegte Timeout abläuft, wird die Socketverbindung zwischen dem Client und dem Server getrennt. Zur Veranschaulichung der Funktionsweise des Severs dient das Beispiel auf der Abbildung 4.3.

4.3 Ergebnis

Alle Methoden des Servers wurden so angepasst, dass ein FTP Client den Zugriff auf das Clippee Dateisystem ermöglicht. Der Benutzer kann im Dateisystem browsen, Dateien und Ordner löschen, Dateien umbenennen, Dateien auf den Server speichern und Dateien vom Server runterladen. Die Integration in den Windows Explorer ist nur in beschränkter Form möglich: es lässt sich mit der IP Adresse des Servers eine Verknüpfung unter der Rubrik `Meine Netzwerkumgebung` erstellen. Dies ist insofern nicht optimal, da einige Programme die Netzwerkumgebung beim `Öffnen` oder `Speichern` einer Datei nicht anzeigen. Das Positive ist, dass der Benutzer keine zusätzliche Client Software benötigt um auf die Daten vom Clippee Dateisystem zuzugreifen, direkten Zugriff aus dem Windows

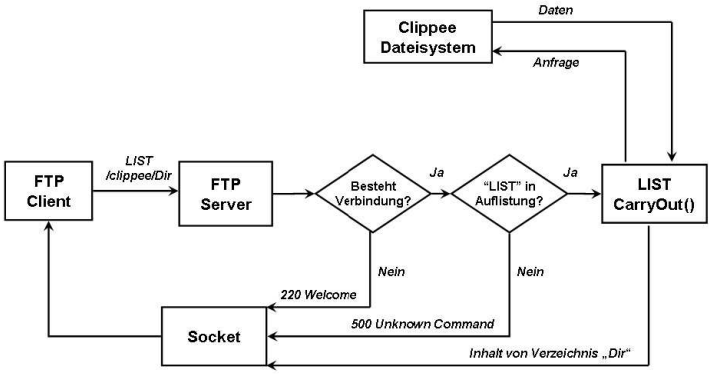

Abbildung 4.3: Das Flussdiagramm einer konkreten FTP Anfrage, welche die Funktionsweise des Servers darstellt.

Explorer hat und die Verknüpfung auch nach einem Neustart erhalten bleibt.

Der Zugriff auf das Clippee Dateisystem über den FTP Server wurde mit einigen verbreiteten FTP Clients getestet und funktioniert sehr zufriedenstellend (siehe Tab. 4.1).

Programm	*Browsen*	*Umbenennen*	*Upload*	*Download*	*Löschen*	*Öffnen*
Windows Exp.	✓	✓	✓	✓	✓	✓
Netscape	-	-*	-*	✓	-*	✓
Opera	✓	-*	-*	✓	-*	-
cuteFTP	✓	✓	✓	✓	✓	-
smartFTP	✓	✓	✓	✓	✓	✓[1]
LeechFTP	✓	✓	✓[2]	✓	✓	✓[1]
WsFTP Pro	✓	✓	✓	✓	✓	✓

Tabelle 4.1: Kompatibilität des Clippee FTP Servers mit einigen verbreiteten Client Programmen.
∗ Funktion wird von den Webbrowsern nicht unterstützt.

4.3.1 Ausblick

Der FTP Server funktioniert sehr zufriedenstellend. Eine mögliche Erweiterung stellt die Authentisierung der Benutzer dar. Diese sollte jedoch vom FTP Server abgekoppelt und in Clippee implementiert werden, so dass sie auch für die SMB Lösung oder andere Clippee Dienste verwendet werden könnte.

[1] Bilder werden als Rohdaten dargestellt.
[2] Leech FTP unterstützt den Upload von Ordnern nicht.

Kapitel 5

Anbindung durch SMB

5.1 Lösungsansatz

Der Lösungsansatz und das Prinzip des SMB Servers sind identisch zur FTP Lösung. Der Unterschied besteht darin, dass der SMB Standardport 139 von Windows schon belegt ist und das SMB Protokoll komplexer ist.

Eine mögliche Umgehung des Problems des Standardports war, einen anderen Port für den Server zu verwenden. Recherchen haben jedoch gezeigt, dass alle Microsoft Produkte standardmässig nur diesen Port verwenden und so auch der Befehl `net use` (Option `Map Network` des Windows Explorers).

5.2 Implementierung

Zuerst muss der Standardport 139 unter Windows XP freigegeben werden, damit der Clippee SMB Server die Anfragen empfangen und bearbeiten kann. Dies erreicht man durch das Deaktivieren von `NetBIOS über TCP/IP` unter den TCP/IP Einstellungen. Damit ist der Port frei und der SMB Server kann darauf zugreifen.

Das hat leider zur Folge, dass der Client die IP Adresse des Servers nicht mehr bekommt. Denn `net use` verwendet beim Aufruf den NetBIOS Namen des Servers, auf welchen dieser durch das Deaktivieren des NetBIOS Services nicht mehr reagiert. Das Problem wird umgangen, indem auf Clientseite eine `LMHOST` Datei[1] mit dem NetBIOS Namen und der IP Adresse des Servers geladen wird. Anstatt die IP Adresse via NetBIOS Service zu ermitteln wird die IP Adresse direkt aus der `LMHOST` Datei übernommen. Eine andere mögliche Lösung dieses Problems ist, einen NetBIOS Service in Clippee einzubinden.

5.2.1 Bestehende SMB Server

Bei der Suche nach bestehenden SMB Servern hat sich gezeigt, dass es fast keine Implementierungen gibt. Der Open Source Server vom Projekt `Samba` [32] ist sehr umfangreich und in der Programmiersprache C geschrieben, weshalb ich mich dazu entschlossen habe, einen neuen SMB Server in Java zu implementieren. Als Hilfe wurde der Code des SMB

[1] Die LMHOST Datei lässt sich unter `TCP/IP Eigenschaften` → `Erweitert` → `WINS` laden.

Clients vom Open Source Projekt `Implementing CIFS in Java` (jCIFS) [31] verwendet. Der Aufbau des Headers und einige Funktionen wurden übernommen, alle anderen Pakete mussten neu implementiert werden. Denn die Anfragepakete unterscheiden sich in Anzahl und Art der Parameter von den Antwortpaketen.

5.2.2 Handshake

Bevor der Client auf das Clippee Dateisystem zugreifen kann, muss die Verbindung erfolgreich aufgebaut werden. Dazu müssen die drei Pakete `Smb_Com_Negotiate`, `Smb_Com_Setup_AndX` und `Smb_Com_Tree_Connect_AndX` implementiert werden. Nach diesem Prozess, dem sogenannten Handshake, wird der freigegebene Ordner des Servers als neues Laufwerk beim Client (in unserem Fall der Windows Explorer) angezeigt.

SMB_COM_NEGOTIATE Es wird ausgehandelt, welche SMB Protokoll Version benutzt wird. Der Client schickt eine Liste der Versionen die er versteht und der Server wählt eine davon aus.

Clippee SMB: Protokollvariante `LM0.12` wird gewählt und der Sicherheitsmodus wird auf Share[2] gesetzt.

SMB_COM_SETUP_ANDX Falls nicht als `Gast` eingeloggt, muss der Benutzer einen gültigen Namen und ein Passwort übermitteln. Bei erfolgreicher Anmeldung erhält der Benutzer eine User ID, welche in jedem nachfolgenden SMB Paket enthalten sein muss.

Clippee SMB: Benutzer wird immer als `Gast` eingeloggt, es wird kein Passwort benötigt. Name des Betriebssystems und des LanManagers werden übermittelt.

SMB_COM_TREE_CONNECT_ANDX Client gibt Pfad des Ordners an, mit welchem er verbunden werden will. Wenn alles korrekt ist (Zugriffsrechte, Pfad existiert ...), bekommt er vom Server eine Tree ID, welche fortan mitgeschickt werden muss, um sich auf diesen Share zu beziehen. Es wird auch angegeben, um welche Art von Service es sich handelt (Datei Service, Drucker, benannte Pipe ...)

Clippee SMB: Es wird immer eine positive Rückmeldung geschickt, eine zufällig generierter Tree ID und dass es sich um einen Datei Service handelt. `Rootordner` ist immer `/clippee/`.

5.2.3 Dateizugriff

Die Verbindung mit dem Clippee SMB Server kann nun erfolgreich durchgeführt werden, und das Dateisystem wird im Windows Explorer als eigenes Laufwerk angezeigt. In einem weiteren Schritt werden die SMB Pakete für die Dateifunktionen implementiert. Das SMB Protokoll hat sich im letzten Jahrzehnt auf über 100 Befehle erweitert, wovon einige Befehle die gleiche Aufgabe erfüllen und viele der Pakete undokumentierte Parameter enthalten [5]. Deshalb wurde zur Implementierung eine Kombination von Referenzdokumenten und Analyse[3] der von Windows verschickten SMB Pakete verwendet. Eine Liste der bereits implementierten SMB Befehle befindet sich im Anhang auf Seite 41.

Der Clippee SMB Server erstellt zu den Anfragen die passenden Antworten, sofern er die Befehle kennt. Die Parameter werden dabei anhand des Clippee Dateisystems oder

[2] Share Sicherheit: Einstellung der Freigabeoptionen der Ordner bestimmen Zugriff auf Daten, Benutzer Sicherheit: Zugriff auf Daten ist abhängig vom angemeldeten Benutzer.

[3] Verwendete Sniffer-Software: Netzwerk Überwachungs Programm `Ethereal`: http://www.ethereal.com.

als fixe Werte gesetzt, z.B. wenn die Parameter undokumentiert sind (siehe Abb. 5.1). Bei einer Änderung des Clippee Dateisystems wird geprüft, ob ein Client ein `Notify` wünscht. Wozu werden Notifies verwendet? Clients schicken für die betrachteten Ordner ein `NotifyRequest` und signalisieren dem Server damit, dass sie bei Veränderungen des Inhaltes informiert werden möchten. Der Server verwaltet die `NotifyRequests` der Clients mit der dazugehörigen FID (File ID) in einer Hashtabelle. Bei Änderungen werden die Clients mit einem `Notify` informiert. So wird sichergestellt dass die Anzeige der Clients immer aktuell ist. Der Zugriff auf die Daten des Clippee Dateisystems ist mittels Windows Explorer möglich.

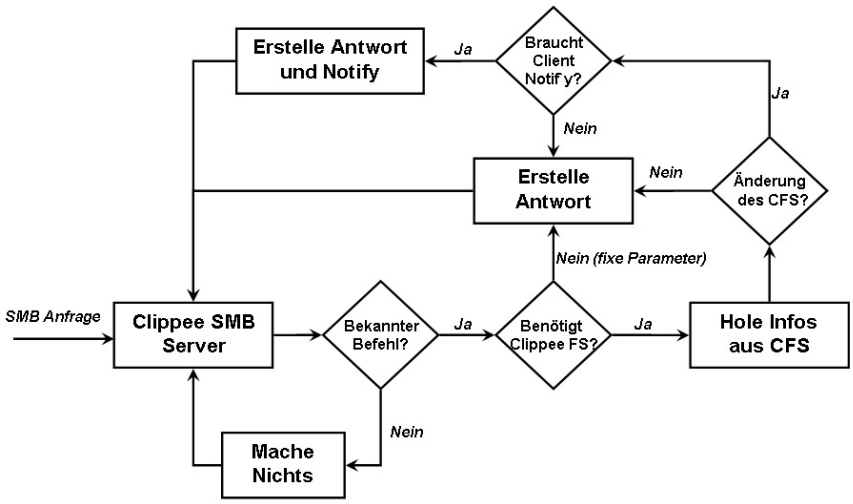

Abbildung 5.1: Das Flussdiagramm zeigt die Abwicklung der SMB Anfragen innerhalb des Servers.

5.3 Ergebnis

Die Anbindung an den Windows Explorer mit einem eigenen Laufwerksbuchstaben funktioniert wunschgemäss, die wichtigsten Funktionen konnten implementiert werden. Diese Lösung ist dem FTP Server vorzuziehen, da die Integration in das Betriebssystem Windows besser vollzogen wird und die Möglichkeiten der Dateimanipulationen besser sind (Verschieben, Erstellen von Dateien, automatisches `Refresh`, ...). Manchen Operationen verursachen Verzögerungen, die entweder vom Programmcode des SMB Servers oder von falsch gesetzten Parametern hervorgerufen werden. Beim Löschen eines Ordners bleibt das Fenster mit dem Fortschrittsbalken des Vorgangs geöffnet, bis der Benutzer dieses manuell schliesst. Trotz des Abbruchs ist der Ordner mitsamt Inhalt gelöscht und wird bei einer manuellen Aktualisierung nicht mehr angezeigt.

5.3.1 Verbesserungen

Der Programmcode kann in verschiedenen Bereichen erweitert und verfeinert werden. Als primäres Ziel gilt es alle momentan implementierten Funktionen zu optimieren, so dass diese effizient und fehlerfrei laufen. Der Server wurde nur mittels eines Threads implementiert, sollte aber erweitert werden sobald er stabil läuft. Ein ThreadPool könnte z.B. zur Anfragenbehandlung verwendet werden.

5.3.2 Erweiterungen

Die Verwaltung der verschiedenen UIDs (User Id), TIDs (Tree ID) und MIDs (Multiplex ID) muss noch implementiert werden. Wenn ein Benutzer sich z.B. ausloggt, dann sollten alle mit ihm verknüpften TIDs und noch nicht abgearbeiteten Anfragen aus dem System gelöscht werden.

Das `Notify` ist auf der Server Komponente implementiert, sollte jedoch direkt in Clippee eingebunden werden. Momentan wird eine Notify Nachricht an den Client geschickt sobald ein Benutzer über den SMB Server eine Veränderung vornimmt. Veränderungen durch eine Shell, eines FTP Servers oder eines SMB Servers auf einem anderen Peer werden jedoch nicht berücksichtigt. Ein anderer Benutzer kann die Daten verändern ohne dass der SMB Client darüber benachrichtigt wird. Auch die Mechanismen zur Verwaltung der UIDs, MIDs und TIDs sollten in Clippee integriert oder damit verbunden werden.

Kapitel 6

Anbindung mit einem Treiber

Aus Zeitgründen konnte dieser Lösungsansatz nicht mehr implementiert werden. Für weiterführende Arbeiten und als Beitrag zum Verständniss, wird in diesem Kapitel der Treiber `Ext2ForXP`[1][26] betrachtet. Dieser baut auf dem RomFS Treiber von Bo Branten [4] und dem FastFAT Treiber von Microsoft auf und ermöglicht den Lesezugriff auf Linux Ext2 Partitionen.

6.1 Lösungsansatz

Die Programmierung eines Treibers benötigt viel Zeit und gute Fachgebietskenntnisse. Deshalb sollte ein Open Source Dateisystem Treiber (z.B. `Ext2ForXP`) an die Clippee Bedürfnisse angepasst werden. Der Clippee Treiber sollte im Kernel-Modus laufen und die Anfragen des E/A Manager an den Clippee FS Manager weiterleiten (siehe Abb.6.1).

Will eine Anwendung auf die Clippee Partition zugreifen, wandelt der `E/A Manager` den `WinAPI32` Aufruf in I/O Request Packets (IRP's) um. Alle benötigten Informationen werden in der IRP Struktur abgelegt. Diese IRP's werden danach an den Treiber weitergeleitet, der sie abarbeitet. Die Resultate der Anfrage werden ebenfalls im IRP abgelegt. Muss der Dateisystemtreiber auf die Daten zugreifen, wird ein neuer IRP (z.B. `IRP_MJ_READ`) erzeugt und an den Clippee FS Manager gesendet, der seinerseits die benötigten Daten aus dem Dateisystem im IRP ablegt.

6.2 Datenstruktur des Ext2 Dateisystemtreibers

Für den Dateisystemtreiber sind vier Datenstrukturen relevant:

- FsdGlobalData (interne Daten des Treibers)
  ⇒ Für jeden Treiber existiert genau ein FsdGlobalData
- VCB (Volume control block)
  ⇒ Für jede gemountete Partition existiert ein VCB
- FCB (File control block)
  ⇒ Für jede Datei existiert maximal ein FCB

[1] Der Treiber wurde uns von der Software Engineering Gruppe (Prof. E.Glatz) der Technischen Hochsule Rapperswil (HSR) zur Verfügung gestellt.

- CCB (Context control block)
  ⇒ Für jede geöffnete Datei existiert genau ein CCB

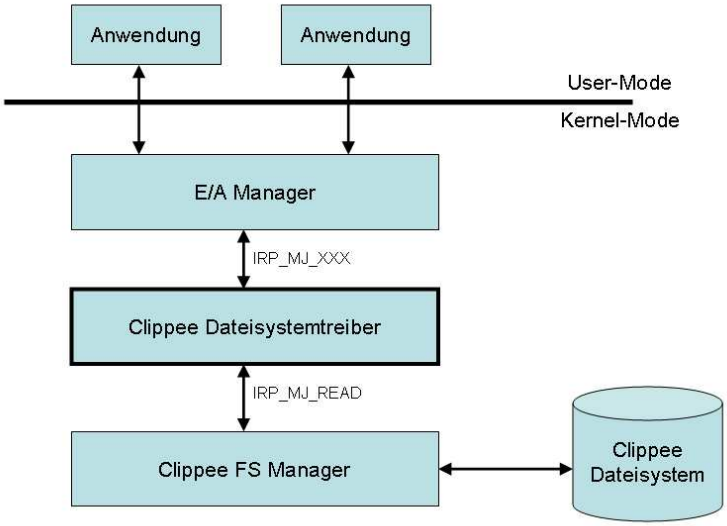

Abbildung 6.1: Der Clippee Dateisystemtreiber wäre die Schnittstelle zwischen E/A Manager und dem Clippee FS Manager.

Kapitel 7

Zusammenfassung

7.1 Clippee Dateisystem

Das verteilte Dateisystem baut auf den Komponenten des Peer-to-Peer Netzwerks `Clippee` auf und bietet die Grundfunktionen (Browsen, Kopieren, Löschen, ...) an. Clippee garantiert die Konsistenz der Daten (siehe 3.1). Aus Zeitgründen und um die Anbindung an Windows nicht zu erschweren wurde der Umfang der Attribute und der Funktionen klein gehalten. Das Dateisystem lässt sich leicht um Attribute erweitern, da die Objekte immer in serialisierter Form gespeichert und weitergeschickt werden. Eine Veränderung der `FS`- und `DirObjects` hat damit keinen Einfluss auf das Clippee Netzwerk.

7.2 Anbindungen

Zum Ermöglichen eines Zugriffs auf das Dateisystem wurden eine Shell, ein FTP Server und ein SMB Server implementiert. Alle drei Dienste können parallel auf einem Peer laufen. Durch das parallele Laufen vom SMB und FTP Server kann die Erreichbarkeit der Daten erhöht werden: Wenn möglich greift der Benutzer mittels SMB auf das Dateisystem zu, sonst kann er immer noch über einen FTP Client (z.B. Webbrowser) auf die Daten zugreifen.

7.2.1 FTP Anbindung

Der FTP Server ist mit den gebräuchlichen Client Programmen kompatibel, und die Operationen werden zuverlässig und schnell durchgeführt. Der Server bietet als Zusatz eine graphische Benutzeroberfläche und viele Einstellungsmöglichkeiten an. Durch die Einschränkungen des Protokolls konnten nicht alle gewünschten Funktionen implementiert werden. Es fehlt das automatische `Refresh` von der Serverseite, das Kopieren/Verschieben innerhalb des Servers und die Anbindung mit einem neuen Laufwerksbuchstaben in das Betriebssystem Windows.

7.2.2 SMB Anbindung

Der SMB Server bietet mehr Dateimanipulationen (Kopieren, Editierung, ...) als der FTP Server an und gliedert sich nahtlos ins Betriebssystem Windows ein. Der Client kann das

Clippee Dateisystem als eigenen Laufwerksbuchstaben im Windows Explorer anmelden und darauf zugreifen. Die Grundfunktionen für die Arbeit mit den Dateien werden vom SMB Server angeboten, sie funktionieren jedoch noch nicht stabil und es entstehen Verzögerungen. Die jetzige Implementierung ermöglicht noch kein `Refresh` und keine sauber abgeschlossene Lösch- und Verschiebungsaktionen mit Ordnern.

Der SMB Server zeigt, dass die wunschgemässe Integration in das Betriebssystem Windows und der Zugriff auf das Dateisystem funktionieren. Aus Zeitgründen ist der Programmcode und die Implementierung verschiedener Pakete noch nicht optimal. Durch eine Fortsetzung der Arbeit am SMB Server sollte eine stabile und leistungsfähige Variante möglich sein.

7.3 Fazit

Im jetzigen Zustand ist der FTP Server dem SMB Server vorzuziehen, da der SMB Server noch instabil und nicht vollständig implementiert ist. Der SMB Server besitzt ein grösseres Potential und sollte weiterentwickelt und letztendlich zur Anbindung für das Clippee Dateisystem unter Windows verwendet werden. Der FTP Server kann danach als zusätzliche Option oder für von SMB nicht unterstützten Plattformen verwendet werden.

7.4 Persönlicher Rückblick

Als sich der SMB Server als bevorzugte Lösung herausstellte, mussten noch mehrere Fragen für den Beginn der Implementierung beantwortet werden: Kann man einen anderen Port als den Standardport 139 verwenden? Wie gibt man den Port 139 unter Windows frei? Muss man den Windows SMB Server abschalten und wie macht man das? Der Zeitaufwand für die Beantwortung dieser Fragen und die Einarbeitung in das Protokoll verzögerten die Implementierung des SMB Servers.

Kapitel 8

Ausblick

8.1 Clippee Dateisystem

Die jetzigen Methoden zur Replikation und Speicherung der Daten sind für Dateisysteme geeignet, welche die Grösse des Hauptspeichers nicht übersteigen. Für grössere Mengen von Daten sollte nur ein Teil der Daten auf einem Peer gespeichert werden um die Speichernutzung zu optimieren und die Netzwerklast zu verringern. Das Replikationsverfahren sollte aber keinen grossen Mehraufwand mit sich bringen, und die Konsistenz der Daten sollte trotz ausfallender Peers erhalten bleiben.

Durch die Aufteilung der Dateien in kleinere Blöcke lässt sich der Speicher und die Netzwerklast weiter optimieren. Der freie Platz auf den Peers wird dadurch besser genutzt und es können kleinere Pakete über das Netzwerk verschickt werden. Weil Clippee zur Speicherung nur den Arbeitsspeicher verwendet, sollten periodische Sicherheitskopien der Daten auf einem persistenten Datenträger (z.B. Festplatte) gemacht werden.

8.2 Anbindung an Windows

Die Möglichkeiten von FTP sind durch die implementierte Lösung so gut wie ausgeschöpft und es bieten sich keine nennenswerte Erweiterungen an.

SMB

Für eine praktische Anwendung des SMB Servers muss die Stabilität und die Geschwindigkeit noch verbessert werden. Ursache für die Instabilität ist entweder der Programmcode des Servers oder falsch oder nicht gesetzte Parameter der Pakete.

Weiter muss die Verwaltung verschiedener Benutzer implementiert und an Clippee angebunden werden. Zur Unterstützung des automatischen `Refreshs`[1] auf der Clientseite wird eine Anbindung der `Notify` Benachrichtigung mit Clippee benötigt. Der Server könnte z.B. die Clippee Nachrichten filtern und bei einer `Delete-` oder einer `Update-Nachricht` betreffend einer Datei, die bei einem Client momentan dargestellt wird, eine Aktualisierung des betrachteten Ordners schicken. So würde sichergestellt dass auch Änderungen ausserhalb des SMB Servers zu einer Aktualisierung führen.

[1] Änderungen des betrachteten Ordners werden automatisch vom Server übermittelt.

Treiber

Das Organisieren des Windows IFS Kits hat viel Zeit beansprucht und dadurch die Implementierung eines Treibers verhindert. Mit dem nun zur Verfügung stehenden Kit und dem Ext2 Dateisystemtreiber für Windows XP lässt sich der Treiber so modifizieren, dass das Clippee Dateisystem in Windows eingebunden werden kann. Aus Zeitgründen konnte nicht abgeschätzt werden ob die Implementierung einer Treiber Lösung Vorteile gegenüber dem SMB Server bringt. Für das Laden des Dateisystemtreibers müsster der Peer auf einem Windows Rechner gestartet werden. Die Benutzer hingegen könnten über das SMB Protokoll auch von einem anderen Betriebssystem aus (z.B. Linux mit Samba) auf das Dateisystem zugreifen.

Literaturverzeichnis

[1] M. Castro G. Cermak R. Chaiken J.R. Douceur J. Howell J.R. Lorch M. Theimer A. Adya, W.J. Bolosky and R.P. Wattenhofer. Farsite: Federated, available, and reliable storage for an incompletely trusted environment. *OSDI 2002*, 2002. 1

[2] T. M. Gil A. Muthitacharoen, R. Morris and B. Chen. Ivy: A read/write peer-to-peer file system. *OSDI 2002: Boston, Massachusetts, USA*, 2002. 1

[3] SNIA (Storage Networking Industry Association). *Common Internet File System (CIFS): Technical Reference*, Januar 2002.

[4] Bo Branten. http://www.acc.umu.se/ bosse/. *Bo Branten: Projects*. 25

[5] CodeFX. http://www.codefx.com/cifs_explained.pdf. *CIFS Explained*, 2002. 22

[6] J. Krebs D. Lüönd, D. Zwirner. http://sourceforge.net/projects/ext2forxp/. *Ext2 Filesystemdriver for Windows XP*, 2002.

[7] Timothy D. Evans. Netbios, netbeui, nbf, nbt, nbipx, smb, cifs networking. http://timothydevans.me.uk/nbf2cifs/book1.html, 1998.

[8] International Organization for Standardization (ISO). http://www.ecma-international.org/publications/files/ecma-st/ecma-119.pdf. *Volume and File Structure of CDROM for Information Interchange*. 9

[9] The Internet Engineering Task Force. http://www.ietf.org/. *IETF: The Internet Engineering Task Force*. 7

[10] Netbios Working Group. http://www.rxn.com/services/rfc/rfc1001.netbios.txt. *Request For Comments: 1001, Protocol Standard For A NetBIOS Service On A TCP/UDP Transport: Concepts and Methods*, März 1987.

[11] Netbios Working Group. http://www.rxn.com/services/rfc/rfc1002.netbios.txt. *Request For Comments: 1002, Protocol Standard For A NetBIOS Service On A TCP/UDP Transport: Detailed Specifications*, März 1987.

[12] PARAGON Software Group. http://www.cdrom-emulator.com/. *CD-ROM Emulator 3.0*. 9

[13] Christopher R. Hertel. http://www.ietf.org/internet-drafts/draft-crhertel-smb-url-06.txt. *SMB File Sharing URI Scheme*, Januar 2004. 3

[14] N. Hranitzky. http://www.norbert.hranitzky.com/jcifs/jcifs_tutorial.html. *jCIFS Tutorial*, August 1998.

[15] S. Menees D. Nichols M. Satyanarayanan R. Sidebotham M. West J. Howard, M. Kazar. Scale and performance in a distributed file system. *TOCS 6*, 1988. 1

[16] M. Satyanarayanan J. Kistler. Disconnected operation in the coda file system. *TOCS 10*, 1992. 1

[17] J. Reynolds J. Postel. http://www.faqs.org/rfcs/rfc959.html. *Request For Comments: 0950, File Transfer Protocol (FTP)*, Oktober 1985. 7

[18] JQ-Data. http://www.jqdata.dk. *JQ-Data FTP-Server*. 17

[19] R. Wattenhofer K. Albrecht, R. Arnold. Clippee: A large-scale client/peer system. *In the Proceedings of the International Workshop on Large-Scale Group Communication, held in conjunction with the 22nd Symposium on Reliable Distributed Systems (SRDS), Florence, Italy*, 2003. 1

[20] A. Bhushan (MIT Project MAC). http://www.faqs.org/rfcs/rfc114.html. *RFC 114 - File Transfer Protocol*, 1971.

[21] Matt. http://www.tuningsoft.com/projects/projects.htm#ext2fsd. *Ext2 File System Driver*. 9

[22] Microsoft. http://www.ntfs.com/. *NTFS - New Technology File System designed for Windows NT, 2000, XP*. 9

[23] Dilip C. Naik(Microsoft) Paul J. Leach (Microsoft). http://www.rxn.com/services/faq/smb/samba.draft-leach-cifs-v1-spec-01.txt. *A Common Internet File System (CIFS/1.0) Protocol: Preliminary Draft*, Dezember 1997.

[24] Dilip C. Naik(Microsoft) Paul J. Leach (Microsoft). http://www.ubiqx.org/cifs/rfc-draft/draft-leach-cifs-v1-spec-02.html. *A Common Internet File System (CIFS/1.0) Protocol: Preliminary Draft*, März 1997. 3

[25] J. Postel. ftp://ftp.isi.edu/in-notes/rfc640.txt. *Request For Comments: 0640, Revised FTP Reply Codes*, Juni 1975.

[26] S. Tweedie R. Card, T. Ts'o. http://e2fsprogs.sourceforge.net/ext2.html. *Ext2fs Home Page*. 9, 25

[27] S. Kleiman D. Walsh B. Lyon R. Sandberg, D. Goldberg. Design and implementation of the sun network file system. *Summer USENIX Proceedings*, 1985. 1

[28] Christopher R.Hertel. *Implementing CIFS, The Common Internet File System*, volume 1st edition. Prentice Hall PTR, August 2003. 3, 4

[29] J. Neefe D. Patterson D. Roselli R. Wang T. Anderson, M. Dahlin. Serverless network file systems. *15th SOSP*, 1995. 1

[30] Christopher R. Hertel (Samba Team). http://www.potaroo.net/ietf/old-ids/draft-crhertel-smb-url-00.txt. *SMB Filesharing URL Scheme*, April 2001.

[31] Samba Team. jcifs: Implementing cifs in java. http://jcifs.samba.org/. 22

[32] Samba Team. Samba web page. http://www.samba.org/. 3, 21

[33] DAEMON Tools. http://www.daemon-tools.cc/dtcc/portal/portal.php. *DAEMON Tools 3.46*. 9

[34] S. Tweedie. http://seclists.org/lists/linux-kernel/2002/oct/att-0520/ext3.txt. *Ext3 Filesystem*. 9

Anhang A

Installieren und Ausführen

A.1 Installieren

Der Sourcecode von Clippee befindet sich auf der CD im Ordner `Source/clippee/`. Damit Logdateien erstellt werden können, sollten die Daten auf die Festplatte kopiert werden. Zum Start von Clippee, FTP Server und SMB Server kann die Klasse `PeerMain` aus dem Paket `clippee.main` verwendet werden.

A.2 Ausführen

A.2.1 Start des ersten Peers

Die Peers werden mit `PeerMain` aus dem Paket `clippee.main` gestartet. Der erste Peer wird mit einer beliebigen Port Nummer als Parameter gestartet. Mit jedem Peer, so auch mit dem ersten, wird als Schnittstelle zum Dateisystem ein Filesystem (FS) Manager gestartet . Der erste Peer erstellt das `FSObject /clippee/` mit dem Key “777”, welcher die Wurzel des Dateisystems repräsentiert. Die Wurzel kann nicht gelöscht werden.

Aufrufbeispiel für den ersten Peer:

```
java -cp clippee.main.PeerMain 6666
```

A.2.2 Weitere Peers anmelden

Weitere Peers können an bestehende Peers angemeldet werden. Die neuen Peers erhalten alle Daten und starten einen FS Manager. Die nötigen Parameter zur Anmeldung sind: die Portnummer des neuen Peers (beliebig), IP Adresse eines Peers des Netzwerkes und dessen Portnummer.

Aufrufbeispiel für einen weiteren Peer:

```
java -cp clippee.main.PeerMain 7777 IP_Adresse 6666
```

A.2.3 Shell starten

Die Shell wird durch den Buchstaben `X` in der Kommandozeile eines Peers gestartet. Die verschiedenen Befehle und deren Bedeutung sind auf der Seite 35 aufgelistet. Weitere Befehle, welche aus der Kommandozeile des Peers gebraucht werden können:

- `P`: Auflistung aller angemeldeten Peers mit Portnummer und IP Adresse.
- `C`: Auflistung aller angemeldeten Clients mit IP Adresse.
- `D`: Auflistung aller Datenobjekte mit Key und Versionsnummer.
- `S`: Shutdown: PeerMain wird beendet.

A.2.4 FTP Server starten

Der FTP Server wird beim Start eines Peers durch den Aufruf des zusätzlichen Parameters `-ftpserver` gestartet. Für die Benutzung der grafischen Oberfäche wird der Parameter `-ftpgui` verwendet.

Aufruf ohne grafische Benutzeroberfläche:
```
java -cp clippee.main.PeerMain 6666 -ftpserver
```
Aufruf mit grafische Benutzeroberfläche:
```
java -cp clippee.main.PeerMain 6666 -ftpserver -ftpgui
```

A.2.5 SMB Server starten

Der FTP Server wird beim Start eines Peers durch den Aufruf des zusätzlichen Parameters `-smbserver` gestartet.

Aufrufbeisipiel :
```
java -cp clippee.main.PeerMain 6666 -smbserver
```

Port 139 freigeben

Damit der Server auf den Port 139 zugreifen kann, muss der Service `NetBIOS over TCP/IP` von Windows abgeschaltet werden. Zusätzlich muss eine `LMHOST` Datei beim Client geladen werden, damit die IP Adresse des Server NetBIOS Namen bekannt ist. Die beiden Optionen befindet sich in:

```
Control Panel/Network Connections/Local Area Connection
→ Eigenschaften
→ Internet Protocol (TCP/IP)
→ Advanced
→ WINS
```

Die LMHOST Datei

Jede Linie der `LMHOST` Datei enthält eine Zuweisung der IP Adresse zu einem NetBIOS Namen. Die Datei für die Zuweisungen der IP Adressen der beiden Computer `wllap8` und `wllap12` würde z.B. so aussehen:

```
129.132.130.243 wllap8
129.132.130.242 wllap12
```

A.3 Shell Befehle

Mit einem `X` in der Kommandozeile des Peers wird die Shell gestartet. `/clippee/` stellt die Wurzel des Dateisystems dar und ist Präfix aller absoluten Pfade (z.B. `/clippee/Dir1/datei.txt/`).

LS Auflistung des Ordnerinhalts. Äquivalent zum Befehl `ls` von Linux und `dir` von der Windows DOS Eingabeaufforderung.

CD Wechseln des Ordners. Als Eingabe kann entweder ein relativer Pfad, absoluter Pfad oder ".." für den übergeordneten Ordner eingegeben werden.

RENAME Umbenennen einer Datei oder eines Ordners. Es wird der alte und der neue Name benötigt.

DEL Löschen einer Datei oder eines Ordners. Der Ordner wird ohne Rückfrage mit dem gesamten Inhalt gelöscht.

COPY Kopiert eine Datei oder einen Ordner in einen anderen Ordner. Der Name des zu kopierenden Objektes wird als relativer Pfad und der Zielordner als absoluter Pfad angegeben.

MOVE Verschiebt eine Datei oder einen Ordner in einen anderen Ordner. Die Eingabeparameter sind die gleichen wie bei dem `COPY`.

READ Liest ein Verzeichnis von der lokalen Festplatte in das Clippee Dateisystem ein. Der erste Parameter bestimmt das einzulesende Verzeichnis (absoluter Pfad), der zweite Paremeter bestimmt wo es im Clippee Dateisystem gespeichert werden soll.

WRITE Funktioniert wie das READ, nur dass jetzt ein Clippee Verzeichnis auf die lokale Festplatte gespeichert wird.

CHECK Ermittelt die Anzahl von Dateien, Ordnern und den benötigten Speicherplatz eines Ordners. Als Eingabeparameter wird der absolute Pfad des Ordners eingegeben.

EXIT Die Shell wird verlassen. Man befindet sich wieder in der Kommandozeile des Peers.

Anhang B

FTP Implementierung

B.1 Implementierte FTP Befehle

Auf den folgenden Seiten befinden sich Tabellen der implementierten FTP Befehle und deren Funktionen. Die Befehle wurden in die drei Kategorien `Zugriffs-`,`Transfer-` und `Servicebefehle` unterteilt.

FTP Zugriffs Befehle	
Befehl	*Funktion*
USER	*User Name* Beinhaltet den Benutzernamen, und wird vom Server mit Hilfe des Passwortes zur Authentifizierung verwendet. Als `Anonymous` braucht man normalerweise kein Passwort und hat Zugriff auf den öffentlichen Bereich.
PASS	*Password* Server überprüft, ob der Benutzer mit dem dazugehörigen Passwort Zugriffsrecht hat.
CWD	*Change Working Directory* Der Benutzer wechselt mit diesem Befehl den aktiven Ordner innerhalb des Dateisystems. Als Parameter wird der gewünschte Pfad des Ordners mitgegeben.
CDUP	*Change To Parent Working Directory* Ist ein Spezialfall des Befehles `CWD`. CDUP wird verwendet, da die Betriebssysteme für den übergeordneten Ordner verschiedene Syntaxen verwenden.
REIN	*Reinitialize* Offenstehende Aktionen werden verworfen, noch aktive Datenübertragungen werden abgeschlossen. Der Zustand entspricht dem nach der Anmeldung.
QUIT	*Logout* Der Benutzer wird abgemeldet und die Verbindung abgebrochen.

Tabelle B.1: Implementierte Zugriffs Befehle des FTP Protokolls.

FTP Transfer Befehle	
Befehl	*Funktion*
PORT	*Data Port* Wird verwendet um Imformationen eines neu kreierten Datenkanals an Client zu schicken. Beinhaltet die IP Adresse (32 Bit) des Servers und die Portadresse (16 Bit) z.B. PORT h1,h2,h3,h4,p1,p2 h: IP Adresse, p: Portadresse
PASV	*Passive* Teilt dem Server mit auf eine Datenverbindung des Clients zu warten, anstatt selber eine zu initialisieren.
TYPE	*Representation Type* Bestimmt die Art der übertragenden Daten (A: ASCII, I: Bytestream)

Tabelle B.2: Implementierte Transfer Befehle des FTP Protokolls.

FTP Service Befehle	
Befehl	*Funktion*
RETR	*Retrieve* Die angegebene Datei wird über den Datenkanal an den Client geschickt.
STOR	*Store* Die angegebene Datei im Pfadnamen wird auf dem Server als Datei gespeichert. Wenn die Datei bereits besteht, wird sie mit den neuen Daten überschrieben, falls sie nicht existiert wird eine Datei erstellt.
ALLO	*Allocate* Wird von einigen Servern verwendet, um zu testen ob auf dem Server noch genügend Speicherkapizität für die folgende `STOR` oder `APPE` Aktion vorhanden ist.
RNFR	*Rename From* Gibt den Pfadnamen der Datei an, welche umbenannt werden soll. Muss direkt von einem `RNTO` gefolgt werden.
RNTO	*Rename To* Enthält den Namen der umzubenennenden Datei. (WS FTP Pro verwendet diesen Befehl zur Verschiebung von Dateien)
ABOR	*Abort* Der vorhergehende Befehl soll abgebrochen werden. Wenn beim vorhergehenden Befehl Daten transferiert wurden, soll der Transfer abgebrochen und der Datenkanal geschlossen werden.
DELE	*Delete* Die angegebene Datei im Pfadnamen wird gelöscht.
RMD	*Remove Directory* Der angegebene Ordner (absoluter Pfad) oder Unterordner (relativer Pfad) wird mit dem kompletten Inhalt gelöscht.
MKD	*Make Directory* Der angegebene Ordner (absoluter Pfad) oder Unterordner (relativer Pfad) wird erstellt.
PWD	*Print Working Directory* Der Pfad des momentan aktiven Ordners wird an den Client geschickt.
LIST	*List* Wenn der Pfadname einen Ordner darstellt, wird eine Liste der enthaltenen Dateien und Ordner übermittelt. Wenn es sich um eine Datei handelt, werden Dateiinformationen geschickt.
SYST	*System* Es werden Informationen zum Betriebssystem, auf welchem der Server läuft, gegeben.
HELP	*Help* Der Server wird veranlasst nützliche Informationen über den Status der Implementierung zu übermitteln. In unserem Beispiel wird eine Liste der implementierten Befehle übermittelt.
NOOP	*No Operation* Hat keine Auswirkung auf andere Befehle oder Parameter. Der Server schickt eine OK Antwort zurück. Dient zur Aufrechterhaltung einer Verbindung und zu testen ob Server noch erreichbar ist.

Tabelle B.3: Implementierte Service Befehle des FTP Protokolls.

Anhang C

SMB Implementierung

C.1 Implementierte SMB Befehle

Name des Befehls	***Byte Code*** *(Dezimal)*
SMB_COM_DELETE_DIRECTORY	0x01 (1)
SMB_COM_CLOSE	0x04 (4)
SMB_COM_DELETE	0x06 (6)
SMB_COM_RENAME	0x07 (7)
SMB_COM_WRITE	0x0B (11)
SMB_COM_ECHO	0x2B (43)
SMB_COM_READ_ANDX	0x2E (46)
SMB_COM_WRITE_ANDX	0x2F (47)
SMB_COM_TRANS2 *(SubCommands in Tab. C.2)*	0x32 (50)
SMB_COM_TREE_DISCONNECT	0x71 (113)
SMB_COM_NEGOTIATE	0x72 (114)
SMB_COM_SETUP_ANDX	0x73 (115)
SMB_COM_LOGOFF_ANDX	0x74 (116)
SMB_COM_TREE_CONNECT_ANDX	0x75 (117)
SMB_COM_NT_TRANS	0xA0 (160)
SMB_COM_NT_CREATE_ANDX	0xA2 (162)

Tabelle C.1: Implementierte Befehle des SMB Protokolls.

Subcommand	(Byte Code)	***Level Of Interest***	(Byte Code)
FIND_FIRST2	(0x0001)		
		FIND_FILE_BOTH_DIR	(0x0104)
QUERY_FS_INFO	(0x0003)		
		FS_INFO_ALLOCATION	(0x0001)
		FS_VOLUME_INFO	(0x0102)
		FS_ATTRIBUTE_INFO	(0x0105)
		FULL_FS_SIZE_INFO	(0x03EF)
NT_NOTIFY	(0x0004)		
QUERY_PATH_INFO	(0x0005)		
		FILE_BASIC_INFO	(0x03EC)
		FILE_STANDARD_INFO	(0x03EE)
		FILE_INTERNAL_INFO	(0x03EE)
		FILE_EA_INFO	(0x03EF)
		FILE_STREAM_INFO	(0x03FE)
QUERY_FILE_INFO	(0x0007)		
		FILE_BASIC_INFO	(0x03EC)
		FILE_STANDARD_INFO	(0x03EE)
		FILE_INTERNAL_INFO	(0x03EE)
		FILE_EA_INFO	(0x03EF)
		FILE_STREAM_INFO	(0x03FE)
QUERY_SET_FILE_INFO	(0x0008)		
		FILE_BASIC_INFO	(0x03EC)
		SET_ALLOCATION_INFO	(0x03FB)
		EOF_INFORMATION	(0x03FC)

Tabelle C.2: Implementierte `Subcommands` von `SMB_COM_TRANS2`.