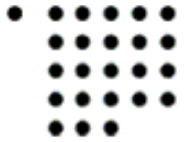

Erweiterung der WebDAV Server Test Suite um das Access Control Protocol – Konzeption und exemplarische Implementierung

mit abschließenden WebDAV Server Leistungstest

Diplomarbeit

zur Erlangung des akademischen Grades

"Diplom-Informatiker (FH)"
im Diplomstudiengang Technische Informatik

an der

Fakultät für Informatik und Ingenieurwissenschaften der Fachhochschule Köln

Themensteller:	Deutsches Zentrum für Luft- und Raumfahrt Einrichtung für Simulations- und Softwaretechnik (SISTEC)
Betreuer:	Abteilungsleiter Dipl. Math. Andreas Schreiber Dipl. Inf. (FH) Markus Litz
Mentor:	Prof. Dr. Hans Ludwig Stahl
Eingereicht von:	Frank Orpel Stammheimer Ring 57 51061 Köln 0221/6086149 franki.o@web.de Matrikel-Nr.: 11032005

Eidesstattliche Erklärung

Ich versichere, die von mir vorgelegte Arbeit selbständig verfasst zur haben. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder nicht veröffentlichten Arbeiten anderer entnommen sind, habe ich als entnommen kenntlich gemacht. Sämtliche Quellen und Hilfsmittel, die ich für die Arbeit benutzt habe, sind angegeben. Die Arbeit hat mit gleichem Inhalt bzw. in wesentlichen Teilen noch keiner anderen Prüfungsbehörde vorgelegen.

Köln, 21. Juli 2007

Frank Orpel

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Einleitung und Aufgabenstellung

1.1 Einleitung

Anfang der neunziger Jahre des letzten Jahrhunderts wurden die Grundlagen für das heutige *World Wide Web* (WWW) geschaffen. Seitdem hat es sich zu einem der größten Informations- und Austauschmedien entwickelt.

Das Offenlegen von Standards, wie das des *Hypertext Transfer Protocols* (HTTP) [RFC2616], haben dazu beigetragen, dass sich das WWW so stark verbreiten konnte. Parallel dazu ist der weltweite Austausch von Informationen zwischen Wissenschaftlern, Unternehmen und anderen Einrichtungen sehr stark angestiegen. Immer häufiger findet die Zusammenarbeit an gleichen Projekten von unterschiedlichen Orten aus statt.

Der zur erfolgreichen Zusammenarbeit nötige Informationsaustausch unterlag in der Vergangenheit jedoch einigen Einschränkungen. Das weit verbreitete HTTP Protokoll gestattete es vorrangig nur lesend auf Informationen zugreifen zu können. Bei der kooperativen Arbeit mussten die Anwender auf andere Dienste zurückgreifen. Für den Austausch von Informationen musste auf andere Protokolle, wie das *File-Transfer-Protokoll* (FTP) [RFC959], zurückgegriffen werden, welche sich vor allem mit dem steigenden Verlangen an der Integration von sichernden Instanzen (Firewalls, Intrusion Detection Systems etc.) zu einer nur schwer überwindbaren Konfigurationshürde auftaten.

Abhilfe schaffte hier das WebDAV [1] Protokoll, welches abkürzend für *Web-based Distributed Authoring and Versioning* steht. Es ist eine Erweiterung des HTTP Protokolls und hat das verteilte, kooperative und kollaborative Arbeiten an gemeinsamen Projekten vereinfacht, indem es das HTTP Protokoll zu einem bidirektional operierenden Protokoll ausgebaut und so auch schreibende Vorgänge ermöglicht hat. Der Einsatz von WebDAV ermöglicht den vollständigen Verzicht auf weitere Protokolle, welche den Datenaustausch betreffen,

und damit eine enorme Minimierung der Konfigurationsansprüche innerhalb von Unternehmen.

Eine Einrichtung, die auf neue und innovative Anwendungen setzt, ist die Einrichtung für *Simulations- und Softwaretechnik* (SISTEC) [2] des *Deutschen Zentrums für Luft- und Raumfahrt* (DLR) [3].

1.1.1 Das Deutsche Zentrum für Luft- und Raumfahrt im Überblick

Das Deutsche Zentrum für Luft- und Raumfahrt (DLR) ist das Forschungszentrum der Bundesrepublik Deutschland für Luft- und Raumfahrt, Energie und Verkehr. Über die eigene Forschung hinaus ist das DLR als Raumfahrtagentur im Auftrag der Bundesregierung für die Planung und Umsetzung der deutschen Raumfahrtaktivitäten zuständig.

Das DLR beschäftigt an seinen 28 Instituten bzw. Test- und Betriebseinrichtungen ca. 5.300 Mitarbeiterinnen und Mitarbeiter und ist an den acht Standorten Köln-Porz, Berlin-Adlershof, Bonn-Oberkassel, Braunschweig, Göttingen, Lampoldshausen, Oberpfaffenhofen und Stuttgart vertreten. Außerdem unterhält das DLR Außenbüros in Brüssel, Paris und Washington D.C.[3]

Die Einrichtung Simulations- und Softwaretechnik (SISTEC) ist auf den DLR-Standorten Köln-Porz und Braunschweig verteilt und gliedert sich in die folgenden Abteilungen:

- Verteilte Systeme und Komponentensoftware und
- Software Qualitätssicherung und eingebettete Systeme

Die Simulations- und Softwaretechnik hat ihre Kompetenz im Bereich des Software Engineering. Die derzeitigen Themenschwerpunkte sind die komponentenbasierte Softwareentwicklung für verteilte Systeme, Software für eingebettete Systeme und die Software-Qualitätssicherung [2].

1.2 Aufgabenstellung und Zielsetzung

Mit dem raschen Wachstum von WebDAV-basierenden Anwendungen und der zunehmenden Datenverwaltung in WebDAV Repositories (mehr dazu in den folgenden Abschnitten), steigt auch die Nachfrage an der Leistung von WebDAV Servern. Jedoch liefern nur einige wenige vorhandene Hilfsprogramme detaillierte Leistungsinformationen von WebDAV Servern.

SISTEC ist an der Entwicklung von WebDAV Client und WebDAV Server Anwendungen mit beteiligt. Deshalb besteht auch hier ein großes Interesse an der Erlangung von solchen detaillierten Leistungsinformationen.

Zwei Hilfsprogramme mit der die Leistungsfähigkeit eines Servers gemessen werden kann, sind die *Litmus-* [4] und die *Prestan [5] WebDAV Server Test Suite*. Dabei prüft Litmus die Konformität der implementierten WebDAV Funktionalitäten und Prestan misst die Performanz in Bezug auf die Verarbeitungsgeschwindigkeit eines WebDAV Servers.

In Zusammenarbeit mit SISTEC, wurden für die hier vorliegende Diplomarbeit zwei Ziele definiert:

1. Da die WebDAV Server Test Suites nicht die WebDAV Funktionalität des *Access Control Protocols (ACP)* [RFC3744] implementiert haben, müssen diese um ACP erweitert werden.

2. Anschließenden gilt es die Leistungsfähigkeit von ausgewählten WebDAV Servern zu testen und zu vergleichen.

Eine Beschreibung von ACP wird im weiteren Verlauf angebracht werden.

1.3 Gliederung der Diplomarbeit

Die vorliegende Diplomarbeit wurde in sieben Kapitel aufgeteilt und beinhaltet folgende Themengebiete:

Kapitel 2: Hintergrund
Dieses Kapitel dient der Einführung in das WebDAV Protokoll und liefert dazu einen historischen Hintergrund. Es wird auf die Grundlagen und Erweiterungen von WebDAV eingegangen und wie sie bis jetzt umgesetzt wurden.

Kapitel 3: Vorstellen der WebDAV Server Test Suites
Soll die Leistungsfähigkeit eines WebDAV Servers bestimmt werden, so kann dies unter Inhilfenahme von Testprogrammen (im nachfolgenden als Test Suites bezeichnet) erfolgen. In Kapitel 3 werden zwei solche Test Suites vorgestellt die einerseits die Konformität des implementierten WebDAV Standards prüfen und andererseits die Leistungsfähigkeit der Verarbeitungs- und Ausgabegeschwindigkeit der Server testen.

Kapitel 4: Konzeption und Implementierung der ACP Erweiterung in den Test Suites
Nachdem die Test Suites vorgestellt wurden, wird in diesem Kapitel auf die Konzeption und die exemplarische Implementierung der ACP Erweiterung in die Test Suites eingegangen. Dabei werden verschiedene Mechanismen vorgestellt.

Kapitel 5: Auswahl der WebDAV Server
Ein Leistungstest für alle Server, bei dem das WebDAV Protokoll implementiert wurde, würde den Rahmen dieser Arbeit sprengen. Deshalb werden in Kapitel 5 einige Kriterien vorgestellt, welche die Auswahl auf wenige Server begrenzt. Außerdem werden die Server und ihre Funktionalitäten kurz vorgestellt und abschließend zusammengefasst.

Kapitel 6: WebDAV Server Test und Auswertung
Nach der ACP Erweiterung der Test Suites, erfolgt in diesem Kapitel der explizite Leistungstest an den zuvor ausgewählten WebDAV Servern. Der Test wird sowohl mit als auch ohne ACP Erweiterung vorgenommen, um im Nachfolgenden Vergleiche zwischen den Testergebnissen anstellen zu können.

Kapitel 7: Zusammenfassung und Ausblicke
In diesem Kapitel werden die Erkenntnis der vorliegenden Arbeit noch einmal zusammengefasst. Außerdem wird es hinsichtlich der weiteren Entwicklung von WebDAV Anwendungen, ein Ausblick auf die Zukunft geben.

Kapitel 2

Hintergrund

2.1 Einführung in WebDAV

Tim Berners-Lee entwickelte 1989 an den Europäischen Kernforschungslaboren (CERN) das World Wide Web (WWW). Er wollte ein System schaffen, das auf Basis des Hypertextes[1] einen weltweiten Austausch, sowie die Aktualisierung von Informationen zwischen Wissenschaftlern ermöglichte. Das WWW sollte sowohl ein lesbares als auch ein editierbares Medium sein. Mit der weiteren Entwicklung wurde es jedoch zu einem *Nur-Lese-Medium*[6].

Zwei Standards haben dazu beigetragen, dass sich das WWW weit verbreiten konnte. Während die *HyperText Markup Language* (HTML)[RFC1866] eine Beschreibungssprache für Dokumente darstellt, die dem Konzept des Hyper-Textes folgen, dient das *Hypertext Transport Protocol* (HTTP)[RFC2616] der Übertragung solcher Dokumente über kommunikationstechnische Infrastrukturen, indem es die Spielregeln der Kommunikation vorschreibt.

Durch diese starke Entwicklung des WWW und dem ständigen Anstieg von globalen und kooperativen Arbeit an gleichen Projekten reichte der gültige HTTP Standard bald nicht mehr aus. Einige Firmen entwickelten proprietäre Ansätze, die zwar auf HTTP aufbauten, aber untereinander nicht kompatibel waren.

1995 kamen während der W3C[2] Konferenz in Boston, einigen Arbeitsgruppen zusammen, die an Umsetzung der ursprüngliche Vision von Tim Berners-Lee und der Problematik des verteilten Editierens von Dokumenten auf Internet-Basis interessiert waren.

So wurde WebDAV 1998 formal durch die *Internet Engineering Task Force*

[1] Hypertext ist eine multi-lineare Organisation von Objekten, deren netzartige Struktur durch logische Verbindungen (so genannte Hyperlinks) zwischen Wissenseinheiten hergestellt wird

[2] das Gremium zur Standardisierung der das World Wide Web betreffenden Techniken

(IETF)[7] bestätigt. Die IETF ist für kurze und mittelfristige Projekte zuständig und entwickelt wichtige Spezifikationen, die in weiterer Folge zu Internet-Standards werden sollen. Die Vorschläge sowie die technischen Details dafür werden in einer Vielzahl von Arbeitsgruppen erstellt.

2.1.1 Was ist Authoring

Bei der netzseitigen Strukturierung durch Autorensystem (Web-Authoring), handelt es sich um einen Prozess, bei dem Inhalte auf Web-Servern erzeugt, aktualisiert und verwaltet werden. Dabei werden auf einem "Standard" Server entsprechend Textdateien in HTML bereitgestellt. In diesen vorformatierten Dateien sind wiederum Verknüpfungen zu anderen Dateien vorhanden. Der von einem Web-Server präsentierte Inhalt kann aber auch Nicht-HTML Inhalte wie Abbildungen, Programme, Musik und Video enthalten[Dus03].

Das Web-Authoring wird in aller Regel aus der Ferne, von einem Client, durchgeführt. Der Web-Server muss dem Client dazu verschiedene Funktionalitäten zu Verfügung stellen, damit er neue Webseiten erstellen und bearbeiten kann. Der Client muss weiterhin in der Lage sein, den aktuellen Inhalt einer Webseite sehen und dem Server neuen Inhalte liefern zu können, um veraltete Inhalte auszutauschen. Clients müssen auch in der Lage sein, veraltete Inhalte zu löschen.

Der Prozess des Authorings gewinnt an zusätzlicher Komplexität, wenn mehrere Web-Autoren gemeinsam an einem Projekt arbeiten. Vor allem parallele Zugriffe auf Dateien können dann zu einer Problematik werden.

2.1.1.1 Die ersten Anfänge

Da es für die ersten Websites noch keine speziellen Hilfsprogramme gab, wurden diese mit einfachen Texteditoren erstellt. Wenn der Web-Autor nicht zufällig direkt an dem Web-Server arbeitete, wurden die fertig erstellten Dateien mit Hilfe des *File Transfer Protocol (FTP)*[RFC959] an den Web-Server übertragen.

Das File Transfer Protocol (FTP) war 1985 standardisiert worden, lange bevor es das WWW überhaupt gegeben hat. Es wurde ausschließlich für den Datentransfer zwischen einem Client und einem Server entwickelt. Das Protokoll bietet keine Mechanismen um die Arbeit von mehreren Autoren zu koordinieren und das Format des Verzeichniseintrags wurde nicht standardisiert.

Die ersten veröffentlichten Web-Autoren Programme waren einfache HTML Editoren, die das Bearbeiten von lokalen HTML Datei vereinfachten. Das "Problem" der Übertragung dieser Dateien zu einem entfernten Server blieb aber weiterhin bestehen.

2.1.1.2 Benutzbare aber eigenständige Software

Die zweite Generation der Web-Authoring Software versuchte dieses Problem zu lösen. Es wurden zum Teil proprietäre Methoden verwendet, um die Dateien auf den Web-Server zu übertragen. Das führte aber zu Inkompatibilitäten zwischen einigen der Web-Autoren Programmen und anderen Software-Applikationen[Dus03].

Vermeer FrontPage welches zu *Microsoft FrontPage*[8] wurde, verwendete so eine proprietäre Methode, um entfernte Web-Seiten zu verwalten. Dies setzt jedoch eine entsprechende Kompatibilität auf der Serverseite voraus, welche mit wenigen Ausnahmen, nur bei Servern garantiert werden kann, welche dem Hause Microsoft entstammen.

Netscape Navigator Gold hat auch die Fähigkeit die Dateien auf den Server mit Hilfe von FTP zu übertragen oder diese über HTTP anzufordern. Für einzelne Web-Autoren ist diese Lösung als ausreichend anzusehen, eignet sich jedoch in keinster Weise zur Verwendung in einem Team. Sowohl FTP als auch HTTP platzieren und überschreiben die Änderungen die von anderen Autoren vorgenommen wurden.

Einige professionelle Web-Entwickler verwenden die Versionsverwaltungssysteme als *Configuration Management* (CM) System für die Verwaltung der Dateien. Zwei der Systeme sind das *Concurrent Versioning System* (CVS) und *Microsoft SourceSafe*.

2.1.1.3 Die dritte Generation der Web Authoring

Die dritte Generation von Web-Authoring Software basierte auf WebDAV und stellte den ersten offenen Standard für Netzstrukturierung durch Autorensystem (Web-Authoring) dar.

Aus dem 1998 angedachten Vorschlag, WebDAV als offenen Standard für das Web-Authoring zu definieren, entwickelte sich diese dritte Generation von Web-Authoring-Software, welche zur entfernten Datenverwaltung nun auf WeDAV setze. Bis 2002 erzielten diese eine breite Interoperabilität und Verfügbarkeit. Die HTTP-Authoring Tools, die nach 2000 eingesetzt wurden, unterstützten WebDAV und konnten so das Speichern von Dateien auf einem Web-Server sicherstellen.

Wie schon in der Einführung erwähnt steht die Abkürzung “WebDAV” für “Web-based Distributed Authoring and Versioning”. Bei der Versionisierung wird automatisch die letzte Version der Webseiten auf dem Server gesichert. Die WebDAV Designer wollten dem Anwender damit die Möglichkeit geben, auf die letzten Versionen von Webseiten zugreifen zu können, um diese wiederherzustellen. Weil die Umsetzung dieser Spezifikation aber sehr kompliziert

war, wurde *Versioning* erst nach drei Jahren in WebDAV standardisiert.

Versioning ist für große, komplizierte Webseiten und für web-basierte Source-Code Verwaltung wichtig, aber es ist nicht für alle Web-Authoring Anwendungen erforderlich. Zum Beispiel unterstützen die Grafikanwendung Photoshop und viele andere Adobe[9] Anwendungen WebDAV, aber nicht Versioning.

WebDAV ist eine Erweiterung von HTTP und kann somit wie HTTP auf die Web-Informationen zugreifen. HTTP erlaubt, Dokumente im Internet zu lesen, aber es kann keine Aufgaben des Web-Authorings übernehmen. WebDAV fügt innerhalb des HTTP-Rahmens neue Funktionalität hinzu, so dass der Austausch von Informationen zwischen den vorhandenen HTTP-Clients und Servern weiter gewährleistet ist.

Nach einigen Jahren der Entwurfsarbeit und nach zahlreichen Diskussionen standardisierte die IETF das WebDAV Protokoll. Die WebDAV Working Group beendete 1998 die Entwurfsarbeiten zu WebDAV und es wurde im Februar 1999 unter den Namen WebDAV als Standardspezifikation akzeptiert und bekam die Identifikation RFC2518[RFC2518].

2.1.2 Ziele des WebDAV Protokolls

Das Ziel der WebDAV Working Group ist: *"to define the HTTP extensions necessary to enable distributed web authoring tools to be broadly interoperable, while supporting user needs"* [1]. Mit WebDAV soll das WWW so erweitert werden, dass es ein beschreibbares und kollaboratives Medium wird. Ziel ist es einen Standard zu schaffen, der den Clients die ferngesteuerte Ausführung von webbasierten Content Authoring Operationen ermöglicht und das die Grundlagen zur Entwicklung interoperierender und kollaborativer Authoring Werkzeuge bereit stellt.

Es gibt auch eine ganze Reihe von Zielen der Nutzer, die mit WebDAV arbeiten. WebDAV wird für das Internet als ein Netzwerk-Dateisystem angesehen oder dient den Zweck der Manipulation von Inhalten eines Dokument Management Systems über das WWW.

Ein weiters Ziel von WebDAV, welches auch das HTTP Protokoll verfolgt, ist eine standardisierte Zugangsschicht (standard access layer) für Repositories zu definieren, wobei HTTP Lesezugriff gewährt, während WebDAV Schreibzugriff auf Ressourcen und Metadaten gibt.

2.1.3 Anwendungsmöglichkeiten von WebDAV

Es gibt eine ganze Reihe von Möglichkeiten für die Anwendung von WebDAV. Nachfolgend sollen hier drei Anwendungsmöglichkeiten kurz erläutert werden.

- verteiltes Editieren (Distributed Editing),
- verteiltes Dokumentenmanagement (Distributed Document Management),
- Versionsverwaltung (Versioning).

2.1.3.1 Verteiltes Editieren

Das verteilte Editieren von Dokumenten beschäftigt sich mit Änderungen an Ressourcen, die nicht durch Operationen auf einem lokalen Dateisystem, sondern durch den Einsatz von HTTP erfolgen.

Öffnen und Schließen von Dokumenten:
Dokumente können als Ressourcen betrachtet werden, die mit unterschiedlichem Inhalt (Text, Grafik...) und einer unterschiedlichen Versionsgeschichte versehen sind. Um diese öffnen zu können, müssen die Ressourcen über entsprechende Meta-Daten verfügen. Serverseitig könnte die Umwandlung der Dokumente in kanonische Formen (HTML, XML) erfolgen, für die Viewer und Browser angeboten werden. Ein Dokumentenmanagementsystem könnte also etwa für das Öffnen einer Ressource die Meta-Informationen und den Inhalt einer Ressource aus dem Speicher laden und eine temporäre Datei für den lesenden Zugriff erzeugen. Für ein tatsächliches Editieren wird dann die Datei im Originalformat zur Verfügung gestellt.

Aus- und Einchecken von Dokumenten:
Zum Öffnen und Schließen gehört auch das Aus- und Einchecken. Das Auschecken benachrichtigt das System darüber, dass das Dokument bearbeitet wird. In der Zeit bis zum Einchecken kann der Benutzer jetzt sicher sein, dass kein anderer das Dokument verändert.

Editieren von Dokumenten:
Nach den Änderungen wird es auf seine ursprüngliche URL [RFC1739] unter der Benutzung von HTTP zurück gespeichert. Beim Einsatz von verschiedenen Sprachvarianten kann eine spezielle Sprache ausgewählt und danach wie oben verändert und gespeichert werden. Bei HTML mit *Server Side Include (SSI)* Direktiven wird neben dem HTML-Code auch der Source-Code der SSI Anweisung übermittelt und kann so angepasst werden.

2.1.3.2 Verteiltes Dokumentenmanagement

Beim verteilten Dokumentenmanagement wird mit Eigenschaften (Properties) von Dokumenten und auch mit der Verwaltung von URL-Hierarchien (vergleichbar mit Ordnern in einem Dateisystem) gearbeitet.

Öffnen und Schließen von Dokumenten:
Sowohl beim Öffnen als auch beim Schließen von Dokumenten entstehen Probleme, wenn zwei Benutzer mit denselben Dokumenten arbeiten. Es muss geregelt werden, wenn ein geöffnetes Dokument von einem anderen Benutzer gelöscht oder verschoben wird. Entweder kann durch das Setzen von Locks der Verschiebevorgang verhindert werden oder die URL wird noch so lange vom Server aufrechterhalten, bis die Änderungen des ersten Benutzers beendet worden sind. Beim Schließen müssen alle laufenden Prozesse sauber beendet worden sein, um den Verlust von Änderungen zu vermeiden.

Erstellen von Dokumenten:
Wie bei Dateisystemen gibt es entweder die Möglichkeit, den exakten Namen und die URL einer neuen Datei anzugeben, oder sich durch die Hierarchie des Web-Servers zu bewegen und eine Ebene für das Dokument auszusuchen (ähnlich dem "Speichern unter" auf lokaler Seite). Zusätzlich dazu ist es zum Beispiel beim Erstellen von HTML-Dateien mit verknüpften Bildern nötig, eine neue Hierarchie erstellen und danach alle mit der HTML-Seite verbundenen Grafiken in das neue Unterverzeichnis ablegen zu können.

Kopieren und Löschen von Dokumenten:
Bei Kopiervorgängen von Dokumenten erfolgt keine unmittelbare Übertragung zum Server, diese werden lokal auf dem Server durchgeführt.

2.1.3.3 Versionsverwaltung

Eine Versionsverwaltung erlaubt das Speichern von wichtigen Versionen eines Dokumentes. Wo normalerweise ein Bearbeiten und Speichern einer Datei die alte Version überschreibt, ermöglicht einem System mit Versionsverwaltung auch den Zugriff auf die Zwischenschritte. Darüber hinaus unterstützt Versionsmanagement eine Zusammenarbeit dadurch, dass zwei oder mehr Benutzer in parallelen Vorgängen Änderungen an demselben Dokument vornehmen können.

2.1.4 Was ist WebDAV?

Im vorangegangen Teil dieser Arbeit wurde schon geklärt, wofür WebDAV benutzen werden kann und warum es entwickelt wurde. Im nachfolgende Abschnitt soll noch einmal erläutern werden, was WebDAV ist und was es nicht ist.

Ist WebDAV auch HTTP?
WebDAV ist eine Erweiterung von HTTP/1.1 und ist in RFC2068 [RFC2068] definiert. Es benutzt bei Versionsanfragen dieselben HTTP/1.1-Kopfzeile an den Server. WebDAV benutzt allen HTTP/1.1 Methoden wie zum Beispiel GET, PUT, und DELETE und kann so nicht von HTTP getrennt werden.

Ist WebDAV eine API?

Weil viele Industriestandards, wie CORBA [10] oder die *Document Management Alliance (DMA)*, Standards sind und das *Application Programming Interface (API)* als Standard haben, wird angenommen, dass WebDAV auch eine API ist. Während APIs jedoch häufig an bestimmte Plattformen gebunden sind, besitzen Protokolle wie WebDAV solche Bindungen in aller Regel nicht.

Ein weiter Unterschied ist, dass APIs normalerweise den Schnittstellenteil zwischen zwei Softwarekomponenten bilden, die auf der gleichen Maschine oder demselben lokale Netz laufen. Heute verfügen viele Sprachen über ferne Prozeduraufrufmechanismen aber trotzdem können APIs Protokolle noch nicht ersetzen. APIs sind selten dafür entworfen worden, Netzlatenzzeiten oder Ähnliches in Betracht zu ziehen oder mit Firewalls zusammen zu arbeiten. WebDAV wurde vom Beginn an als ein Internetprotokoll entwickelt, das mit hohen Latenzzeiten, ausgelasteten Servern, Firewalls und Proxies arbeitet kann.

Ist WebDAV ein Document Management System?

Document Management System werden heutzutage zumeist als großes Software-Pakete angeboten. Bei einigem dieser Systeme ist die WebDAV Unterstützung für die Einkaufsentscheidungen ein wichtiger Faktor. Diese Systeme könnten sowohl Clientsoftware als auch Serversoftware sein, welche mit Hilfsprogrammen ausgestattet ist, um die Dokumenten-Repositories zu verwalten. WebDAV entspricht in diesem Kontext im weitesten Sinne dann lediglich einer Komponente eines komplexen Document Management Systems und verfolgt damit den typischen Protokollcharakter.

Ist WebDAV ein würdiger Nachfolger von FTP?

WebDAV kann im unmittelbaren Vergleich mit FTP einige Vorteile für sich verbuchen und FTP sogar gänzlich ersetzen, wenn es um die Übertragung von Webseiten geht. Für Website-Authoring bietet WebDAV besonders wichtige Vorteile gegenüber FTP an:

Da WebDAV eine Erweiterung von HTTP darstellt, gibt es bezüglich der Adressierung von Ressourcen keine Unterscheidungen.

WebDAV ist viel flexibler bei der Wahl der Sicherheitsoptionen und bietet vergleichbar zu HTTP mehreren Wegen für eine Authentifizierung und Identifizierung von Clients und Servers an.

WebDAV hat eine standardisierte Verzeichnisstruktur. Im Gegensatz dazu können die FTP Verzeichniseinträge irgendwelche Formatierungen annehmen und machen es dem Anwender schwer, zu wissen, wie man die Daten syntaktisch analysieren und sie zeigen kann.

Weiterhin bietet WebDAV einen Sperrmechanismus für einzelne Ressourcen

an, um zu verhindern, dass mehrere Autoren ihre jeweils getätigte Arbeit gegenseitig überschreiben.

Ein weiterer Punkt wäre, dass beim Einsatz von FTP zwei Konfigurationen in einem Unternehmen gepflegt werden müssen, was zu Problemen mit Firewalls etc. führen kann. Würde WebDAV verwendet werden, so wäre lediglich eine Konfiguration einzubringen.

Ist WebDAV ein Standard?

Unter der RFC Nummer (RFC2518) [RFC2518] ist die WebDAV Spezifikation als ein IETF Standard-Dokument genehmigt worden. Die IETF teilt den Standardisierungsverlauf in drei Levels ein und WebDAV hat dabei den ersten vorgeschlagenen Standard erreicht. Zukünftig kann WebDAV zu einem Vorabstandard wie HTTP/1.1 [RFC2616] oder sogar zu einem Internetstandard werden, wie dies zum Beispiel das *Transmission Control Protocol (TCP)* [RFC793] ist.

2.2 Grundlagen von HTTP und die Erweiterungen von WebDAV

2.2.1 Hypertext Transfer Protokoll (HTTP)

Wie schon am Anfang des Kapitels zu erfahren war, ist Tim Berners-Lee ein Mitbegründer des World Wide Webs (WWW). Er und seine Mitstreiter entwickelten 1989 unter anderem das Hypertext-Transfer-Protokoll (HTTP). Es ist ein Protokoll der Anwendungsschicht und baut auf TCP/IP, den Protokollen der Transport- und Netzwerkschicht, auf. Es dient im Wesentlichen zur Übertragung von Dokumenten und Metadaten und ist nicht nur auf Texte beschränkt.

HTTP basiert auf dem Client-Server Modell. Ein HTTP Client öffnet eine Verbindung und sendet eine Anfrage (**Request**) an einen HTTP Server. Der Server gibt eine Rückantwort (**Response**), die die angeforderten Daten (oder einen Fehler) enthält. Mit dem Abschluß der Transaktion wird die Verbindung geschlossen. Eine erneute Anfrage baut eine neue Verbindung auf, ohne zu wissen, welche Befehle zuvor gesendet wurden. Deshalb wird HTTP auch als ein zustandsloses Protokoll genannt, weil jeder Befehl unabhängig vom vorhergehenden Befehl ausgeführt wird. Die Adressierung der Daten findet dabei über **URLs** statt. HTTP wickelt die Interaktion zwischen Client und Server ab und sorgt für die Anpassung der Formate zwischen Client und Server [6] [11].

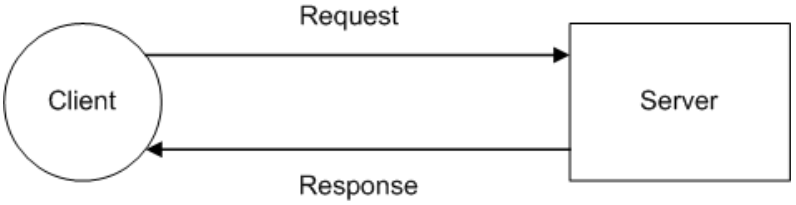

Abbildung 2.1: Client-Server Modell

2.2.1.1 URLs

Die *Uniform Resource Locator (URL)* [RFC1739] ist eine Unterart der *Uniform Ressource Identifier (URI)* [RFC3986] und kann als digitale Adresse einer Ressource, wie die Post-Adresse eines Hausen bzw. einer Person im realen Leben angesehen werden. Beide besitzen spezielle Informationen: eine geographische Adresse hat beispielsweise eine Hausnummer, während eine URL den Namen einer Ressource aufzuweisen hat. Allgemein betrachtet identifiziert die URL eine Ressource in einem Netzwerk eindeutig [Dus03].

Die Struktur einer URL kann wie folgt aussehen:

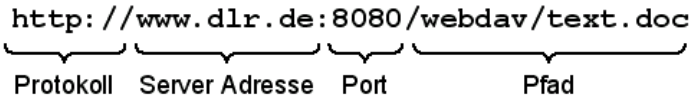

Abbildung 2.2: Beispiel für eine HTTP URL Struktur

2.2.1.2 Nachrichtenstruktur

Wie schon erwähnt, findet der Nachrichten Austausch zwischen Client und Server in einem so genannten Request-Response-Zyklus statt. Erfolgt eine Anfrage eines gewünschten Dokuments vom Client (zB. Web-Browser), mittels des HTTP-Request an den in der URL spezifizierten Web-Server, wird eine TCP/IP-Verbindung aufgebaut, über die der Request erfolgt. Wenn der Web-Server die angeforderte Seite zur Verfügung stellen kann, so wird das Dokument über dieselbe Verbindung durch den HTTP-Response zurückgeschickt. Anschließend wird die Verbindung geschlossen. Müssen weitere Daten übertragen werden (z.B. Bilder, Dateien), wird der gleiche Vorgang wiederholt, bis alle gewünschten Daten beim Web-Browser angekommen sind.

HTTP-Request:
Grundsätzlich besteht ein HTTP-Request aus:

- der *Request Line* mit den Methoden, dem Request-URI und der Protokolversion,
- dem Header,
- einer Leerzeile (Empty Line) und
- dem optionalen Body

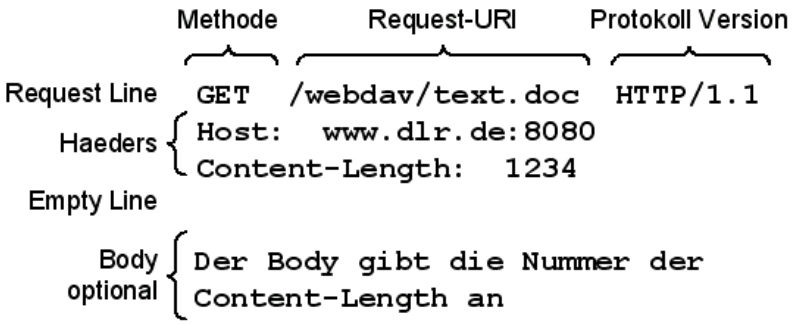

Abbildung 2.3: HTTP-Request

HTTP-Response:
Auf jeden HTTP-Request folgt eine HTTP-Response, die analog zu dem Request aufgebaut ist. Zusätzlich werden hier noch eine Status-Line und eine

Entity-Line eingefügt, in welchen in aller Regel die angeforderten Daten übertragen werden.

Die Struktur des HTTP-Response sieht dann wie folgt aus:

- Status Line mit der Protokollversion, Status Code und Status Description,
- dem Header,
- einer Leerzeile (Empty Line) und
- dem optionalen Body

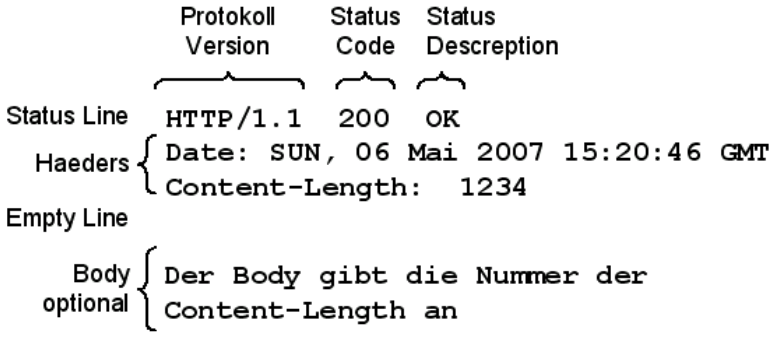

Abbildung 2.4: HTTP-Response

2.2.1.3 HTTP Methoden

Das HTTP-Protokoll (Version 1.1 im RFC2616 [RFC2616]) definiert acht Methoden, die für den Zugriff auf Ressourcen zuständig aber nicht alle für WebDAV relevant sind.

- **OPTIONS:**
  Mit dieser Methode stellt der Client fest, welche Methoden von einer Ressource unterstützt werden. In der Serverantwort wird dazu der Allow Response-Header mit den Namen der möglichen Methoden gesetzt.
- **GET:**
  weist den Server an, die Ressource (Dokument, Graphik) zu schicken.
- **HEAD:**
  weist den Server an, nur den HTTP-Header zu schicken.
- **PUT:**
  Die im Datenteil vorhandenen Daten sollen vom Server unter der angegebenen URL abgespeichert werden. Dies setzt natürlich geeignete Berechtigungen voraus.
- **POST:**
  Erzeugt ein neues Objekt, welches über einen Link zum spezifizierten Objekt in Beziehung gesetzt wird und diesem untergeordnet ist. POST wird z.B. verwendet, um die Eingaben eines Formulars an den Server zu übergeben.
- **DELETE:**
  Mit der DELETE Methode können Ressourcen vom Server entfernt werden
- **TRACE:**
  wird nur für Testzwecke benötigt. Der Server sendet die empfangene Anfrage als Datenteil in der Antwortnachricht zurück.
- **CONNECT:**
  wird von Proxyservern implementiert, die in der Lage sind, SSL-Tunnel zur Verfügung zu stellen.

2.2.2 Die Modifikationen von WebDAV zu HTTP

In HTTP/1.1 werden alle Webobjekte als Ressource bezeichnet, die durch eine Uniform Resource Identifiers (URI) adressiert werden. WebDAV erweitert dieses Datenmodell und führt die *Collection* als neuen Ressourcentypen ein.

- Eine Collection ist eine spezielle Art einer Ressource, die als Container für andere Ressourcen dienen kann.
- Alle Ressourcen haben Eigenschaften (Properties) mit Namen- und Wertepaaren, wobei die Properties nicht unabhängig adressierbar sind und keine eigene URI besitzen.

Daraus kann geschlossen werden, dass die WebDAV Collections mit Verzeichnissen und die andere Ressourcen mit Dateien gleichgesetzt werden können. Demnach kann eine Ressource unter der Betrachtung des WebDAV Kontextes entweder eine Datei oder eine Collection (Verzeichnis) sein, die mit ihren Unter-Ressourcen (Kindern) eine Baum abbilden.

Das WebDAV Protokoll unterstützt nicht nur Operationen an einzelnen Ressourcen. Durch die Spezifikation mittels eines *Depth-Header* können auch Operationen an einem gesamten Baum bzw. an einer Collection mit seinen Kindern durchgeführt werden.

Drei Tiefenwerte (Depth-Values) können verwendet werden:

- Depth 0: Die Operationen werden nur auf die gegebenen Ressourcen angewendet.
- Depth 1: Die Operationen gelten für die gegebenen Ressourcen (Collection) und seinen direkten Kinder.
- Depth infinity: Die Operationen werden auf die gegebenen Ressourcen angewendet, den Collection und allen Kinder die im Baum unter den gegebenen Ressourcen liegen [RFC2518].

2.2.2.1 Name-Space Verwaltung

Die *Name-Space Verwaltung* bietet das entfernte Erstellen und Bearbeiten von Ressourcen (Ressourcen, Collections und deren Ressourcen) an. Dafür erweitert WebDAV die schon erörterten HTTP-Methoden und fügt einen neuen Satz von Methoden hinzu. Die nachfolgende Methodenzusammenfassung, soll darüber einen kleinen Überblick geben.

- **OPTIONS:**
  OPTIONS wurde so erweitert, um festzustellen, welche WebDAV Unterstützung der Server bereitstellt. Dazu wurden zwei Headerzeilen hinzugefügt.

  Der Header führt zum einen die unterstützte WebDAV Klasse auf. Jeder WebDAV Server muss die in RFC2518 [RFC2518] spezifizierte Klassifizierung 1 zwingend unterstützen. Zum anderen zeigt die *Allow*-Zeile alle Methoden auf, die auf eine Ressource angewendet werden können.

  Beispiel:

  »»»Request

```
OPTIONS /test/document.doc HTTP/1.1
Host: www.example.com

.
```

Listing 2.1: Beispiel: Request - OPTIONS

»»»Response

```
HTTP/1.1 200 OK
Date: Mon, 4 Jul 2005 15:30:00 GMT
Server: Apache/1.3.14 (Unix) DAV/1.0.2
DAV: 1,2
Allow: OPTIONS, GET, HEAD, DELETE, TRACE, PROPFIND,
PROPPATCH, COPY, MOVE, PUT, LOCK, UNLOCK

.
```

Listing 2.2: Beispiel: Response - OPTIONS

- **GET und HEAD:**
  GET und HEAD verhalten sich genau so wie beim einfachen HTTP. Sie werden verwendet, um angefragte Ressourcen vom Server herunterladen zu können. Eine Ressource kann dabei eine Webseite, Abbildung oder auch ein Dokument sein.
- **PUT:**
  PUT wird verwendet um eine Datei auf den Server zu übertragen und kann dabei nur auf eine so genannte *Non-Collection-Ressource* angewendet werden. Für Collections ist der nachfolgend erklärte Befehl MKCOL zuständig.
- **DELETE:**
  Die DELETE Methode wird wie auch beim einfachen HTTP dazu verwendet, um Ressourcen - egal welcher Art - zu entfernen.

- **MKCOL:**
  Mit MKCOL (*Make COLlection*) werden Ressourcen des neuen Typs *Collection* erzeugt. Wichtig ist hier bei, dass mit der MKCOL Methode eine neue Collection nur dann erzeugt wird, wenn alle übergeordneten Collections bereits existieren.
- **MOVE:**
  Die MOVE Methode wird verwendet, um Ressourcen umzubenennen oder diese zu einem anderen Standort zu verschieben und dabei alle zugehörigen Eigenschaften zu berücksichtigen.
- **COPY:**
  Mit der COPY Methode werden die Ressourcen an das Ziel kopiert, welches in der *Destination*-Zeile des Request-Headers der vom Client gesendeten Anfrage steht. Die *Overwrite*-Zeile des Request-Header legt fest, ob eine bestehende Ressource am Ziel überschrieben wird. Beim Kopieren von Collections werden die enthaltenen Ressourcen standardmäßig mit kopiert. Um nur die Collection selbst zu kopieren und die enthaltenen Ressourcen dabei zu ignorieren, wird die *Depth*-Zeile des Request-Headers auf den Wert 0 gesetzt.

  Beispiel:

  »»»Request

```
COPY /test/document.doc HTTP/1.1
Host: www.example.com
Depth: 0
Destination: http://www.example.com//test/copy/document.doc
Overwrite: T

.
```

Listing 2.3: Beispiel: Request - COPY

»»»Response

```
HTTP/1.1 201 Created
Date: Mon, 4 Jul 2005 15:30:00 GMT
Content-Length: 0

.
```

Listing 2.4: Beispiel: Response - COPY

2.2.2.2 Verwalten von Dokumenteneigenschaften (Metadaten)

Metadaten sind Daten, die Informationen über andere Daten enthalten. Diese werden auch als **Properties** (Eigenschaften) bezeichnet und lassen sich in zwei Gruppen einteilen:

- **Dead Properties:**
  Dead Properties sind jene Metadaten, die hauptsächlich durch den Client oder den Benutzer verwaltet werden. In diese Gruppe fallen beispielsweise der Titel des Dokuments oder die Schlüsselworte. Der Name wurde gewählt, um auszudrücken, dass diese Properties, wenn sie einmal im System erfasst worden sind, nicht mehr geändert werden, außer sie werden explizit durch den Benutzer modifiziert. Auch werden diese durch den Server nicht ausgelesen und auf Gültigkeit überprüft.
- **Live Properties:** Live Properties Metadaten, die im Gegensatz zu den Dead Properties durch den Server bzw. das System erfasst und verwaltet werden. Beispiele hierfür sind das Datum der letzten Änderung, sowie die Größe des Dokuments.

Um die Properties lesen und schreiben zu können, stellt das WebDAV-Protokoll zwei Methoden zu Verfügung.

PROPFIND

Die PROPFIND Methode liest die Properties einer Ressource aus und bekommt standardmäßig von ihr alle Informationen zu der Ressource zurückgeliefert. Ein Client kann aber auch nur bestimmte Properties oder nur die Namen aller Properties anfordern. Die genaue Anfrage wird mit Hilfe von XML im Datenteil der Nachricht formuliert.

Beispiel:

»»»Request

```
PROPFIND /test/level1/ HTTP/1.1
Host: www.example.com
Content-Type: text/xml; charset="UTF-8"
Content-Length: xxxx
Date: Mon, 4 Jul 2005 16:31:12 GMT

<?xml version="1.0" encoding="utf-8" ?>
<D:propfind xmlns:D="DAV:">
 <D:prop>
  <D:resourcetype/>
  <D:getlastmodified/>
 </D:prop>
</D:propfind>

.
```

Listing 2.5: Beispiel: Request - PROPFIND

»»»Response

```
HTTP/1.1 207 Multi-Status
Content-Type: text/xml; charset="UTF-8"
Content-Length: 638
Date: Mon, 4 Jul 2005 17:13:14 GMT

<? xml version="1.0" encoding="utf-8"?>
<D:multistatus xmlns:D="DAV:">
<D:response>
 <D:href>http://www.example.com/test/level1/</D:href>
```

```
 <D:propstat>
  <D:prop>
   <D:resourcetype><collection/></D:resourcetype>
   <D:getlastmodified>Mon, 4 Jul 2005 09:30:05 GMT</D:getlastmodified>
  </D:prop>
 </D:propstat>
</D:response>

<D:response>
 <D:href>http://www.example.com/test/level1/dir/</D:href>
 <D:propstat>
  <D:prop>
   <D:resourcetype><collection/></D:resourcetype>
   <D:getlastmodified>Son, 3 Jul 2005 17:31:43 GMT</D:getlastmodified>
  </D:prop>
 </D:propstat>
</D:response>
.............
.............
.............
</D:multistatus>

.
```

Listing 2.6: Beispiel: Response - PROPFIND

PROPPATCH

Die PROPPATCH Methode ist das Gegenstück zur PROPFIND Methode. Mit dieser können die Properties einer Ressource gesetzt und gelöscht werden. Dabei ist das Setzen und Löschen mehrerer Properties gleichzeitig möglich. Wie PROPFIND nutzt auch PROPPATCH XML zur Formulierung der Anfrage.

Beispiel:

»»»Request

```
PROPPATCH /test/level1/ HTTP/1.1
Host: www.example.com
Content-Type: text/xml; charset="UTF-8"
Content-Length: xxxx
Date: Mon, 4 Jul 2005 16:31:12 GMT

<?xml version="1.0" encoding="utf-8" ?>
<D:propertyupdate xmlns:D="DAV:" xmlns:nstu="http://www.example.com">
 <D:set>
  <D:prop><nstu:linked>false</nstu:linked></D:prop>
 </D:set>
 <D:remove>
  <D:prop><nstu:temp/></D:prop>
 </D:remove>
</D:propertyupdate>

.
```

Listing 2.7: Beispiel: Request - PROPPATCH

»»»Response

```
HTTP/1.1 207 Multi-Status
Content-Type: text/xml; charset="UTF-8"
Content-Length: xxxx
Date: Mon, 4 Jul 2005 17:13:14 GMT

<?xml version="1.0" encoding="utf-8" ?>
<D:multistatus xmlns:D="DAV:" xmlns:nstu="http://www.example.com">
 <D:response>
 <D:href>http://www.example.com/level1/</D:href>
  <D:propstat>
```

```
    <D:prop><nstu:linked/></D:prop>
    <D:status>HTTP/1.1 403 Forbidden</D:status>
   </D:propstat>
 </D:response>
.............
.............
.............
</D:multistatus>

.
```

Listing 2.8: Beispiel: Response - PROPPATCH

2.2.2.3 Sperren von Dokumenten (Locking)

Ein wichtiger Aspekt beim verteilten Authoring ist die Regelung des konkurrierenden Zugriffs auf eine Ressource. Dazu bietet WebDAV einen Mechanismus an, der den Zugriff auf einer Ressource serialisiert zulässt. Mit Hilfe der LOCK Methode, garantiert der Server, dass nur ein Benutzer die Ressource bearbeiten kann. Es kommen dabei zwei Arten des Locking von Ressourcen zur Anwendung.

- **Shared Locking:**
  Das *Shared Locking* erlaubt den gleichzeitigen Zugriff von mehreren Benutzern auf einer Ressource. Unter Ausnutzung eventueller Absprachen zwischen den konkurrierenden und kooperierenden Benutzer ist so ein gleichzeitiges Bearbeiten einer Ressource möglich.
- **Exclusive Locking:**
  Am häufigsten wird aber das *Exclusive Locking* verwendet. So ist sichergestellt, dass nur der Benutzer, der den Lock auf eine Ressource angelegt hat, auch eine Bearbeitung vornehmen kann. Wollen andere Benutzer diese Ressource bearbeiten, so müssen sie warten, bis der Besitzer des Locks diese durch ein Unlock wieder entfernt hat oder ein Timeout die Ressource wieder freigibt.

Lock Tokens

Um eine Lock zu identifizieren, wird ein Lock-Token verwendet, das durch eine URI repräsentiert wird. Lock-Token URIs müssen einzigartig über alle Ressourcen sein, um die Mehrdeutigkeit zu verhindern.

LOCK

Mit der Methode LOCK kann die im Request-URI angegebenen Ressource reserviert werden, so dass sie für andere Benutzer nicht mehr änderbar ist.

Falls der Request erfolgreich bearbeitet werden konnte, erhält der Client ein Lock-Token zurück, das er beim Aufheben des Locks wieder vorweisen muss.

Beispiel:

»»»Request

```
LOCK /workspace/webdav/proposal.doc HTTP/1.1
Host: webdav.sb.aol.com
Timeout: Infinite, Second-4100000000
Content-Type: text/xml; charset="utf-8"
Content-Length: xxxx
Authorization: Digest username="ejw",
realm="ejw@webdav.sb.aol.com", nonce="...",
uri="/workspace/webdav/proposal.doc",

<?xml version="1.0" encoding="utf-8" ?>
<D:lockinfo xmlns:D='DAV:'>
 <D:lockscope><D:exclusive/></D:lockscope>
 <D:locktype><D:write/></D:locktype>
 <D:owner><D:href>http://www.ics.uci.edu/~ejw/contact.html</D:href></D:owner>
</D:lockinfo>

.
```

Listing 2.9: Beispiel: Request - LOCK

»»»Response

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset="utf-8"
Content-Length: xxxx

<?xml version="1.0" encoding="utf-8" ?>
<D:prop xmlns:D="DAV:">
 <D:lockdiscovery>
  <D:activelock>
   <D:locktype><D:write/></D:locktype>
   <D:lockscope><D:exclusive/></D:lockscope>
   <D:depth>Infinity</D:depth>
    <D:owner>
     <D:href>http://www.ics.uci.edu/~ejw/contact.html</D:href>
    </D:owner>\\
   <D:timeout>Second-604800</D:timeout>
   <D:locktoken>
    <D:href>opaquelocktoken:e71d4fae-5dec-22d6-fea5-00a0c91e6be4</D:href>
   </D:locktoken>
  </D:activelock>
 </D:lockdiscovery>
</D:prop>

.
```

Listing 2.10: Beispiel: Response - LOCK

UNLOCK

Mit der Methode UNLOCK kann die im Request-URI angegebene Ressource wieder freigegeben werden, so dass sie für andere Benutzer wieder änderbar ist. Der Client muss bei dem Request das Lock-Token wieder übergeben, das ihm beim LOCK-Request erteilt wurde.

Beispiel:

»»»Request

```
UNLOCK /workspace/webdav/info.doc HTTP/1.1
Host: webdav.sb.aol.com
Lock-Token: <opaquelocktoken:a515cfa4-5da4-22e1-f5b5-00a0451e6bf7>
Authorization: Digest username="ejw",
realm="ejw@webdav.sb.aol.com", nonce="..."
uri="/workspace/webdav/proposal.doc",


.
```

Listing 2.11: Beispiel: Request - UNLOCK

»»»Response

```
HTTP/1.1 204 No Content


.
```

Listing 2.12: Beispiel: Response - UNLOCK

In diesem Beispiel wird der durch den Lock Token

- “opaquelocktoken:a515cfa4-5da4-22e1-f5b5-00a0451e6bf7”

identifiziertes Lock von der Ressource
http://webdav.sb.aol.com/workspace/webdav/info.doc erfolgreich entfernt.

2.3 Weitere WebDAV Standards

Nachdem die IETF Arbeitsgruppe die Anforderungen für das WebDAV Protokoll festgelegt hatte, welche im RFC2291 [RFC2291] beschrieben wird, wurden die Aufgaben auf drei Gruppen aufgeteilt. Diese drei Gruppen sind die *WebDAV Working Group*, die *DASL Working Group* und die *DeltaV Working Group*. Die WebDAV Working Group übernimmt dabei alle allgemeinen Aufgaben und ist für die grundlegende Pflege der Standardspezifikation von WebDAV verantwortlich [1].

Alle weiteren WebDAV-Spezifikation werden auf die drei Gruppen aufgeteilt und werden im nachfolgenden Abschnitt näher erläutert.

2.3.1 Versionisierung von Ressourcen (DeltaV)

Die Versionisierung von Ressourcen übernahm die DeltaV Working Group. Diese Spezifikation gehört zu einer der Hauptkomponenten von WebDAV. Mit der Veröffentlichung des RFC3253 [RFC3253] wurde die WebDAV-Spezifikation um Versionskontrolleigenschaften erweitert und somit komplettiert, da RFC2518 [RFC2518] nur das verteilte Bearbeiten von Ressourcen abdeckt.

Die Spezifikation, welche auch als DeltaV bezeichnet wird, definiert neue Methoden, Header und Eigenschaften, um Versionskontrolle für entfernte Ressourcen nutzbar zu machen. Dabei werden folgende Funktionen geboten: checkout, check-in (inklusive Kommentare), Version Graph History, durchsuchen von alten Versionen, automatische Versionierung für Clients, die kein Versioning unterstützen, und einfache sowie high-level Konfigurationsoperationen.

2.3.2 DAV Searching und Locating (DASL)

Die DASL Working Group arbeitet an WebDAV-Spezifikation für das entfernte Suchen in Resourcen und Metadaten (remote searching).

Es kann nach Ressourcen gesucht werden, von denen eine Eigenschaft oder ein Wert bekannt ist. Weiterhin kann auch nach einer Zeichenkette in einer Ressource gesucht werden. Der Bereich der Suche kann beschränkt werden, entweder auf einen ganzen Server, eine Hierarchie von Ressourcen, eine Collection von Ressourcen oder eine einzelne Ressource.

2.3.3 Access Control Protocol (ACP)

Eine Untergruppe der Working Group ist die *Access Control Protocol Subgroup*. Diese Gruppe arbeitet an der Entwicklung der entfernten Zugangskon-

trolle (Access Control).

Da die Implementierung der ACP Erweiterung in einer Test Suite einen wesentlichen Teil dieser Arbeit darstellt, wird im Nachfolgenden näher auf das Access Control Protocol eingegangen. Dabei beziehen sich alle Funktionalitäten, auf die im RFC3744 [RFC3744] beschrieben Spezifikationen des Access Control Protocols.

Mit Hilfe der WebDAV Access Control Erweiterung sollen dem WebDAV Server kompatible Mechanismen zur Verfügung gestellt werden, die die Zugangskontrolle (Access Control) auf verschiedenste Ressourcen regelt. Dabei greift WebDAV zugrunde liegende Prinzipien der Zugangskontrolle auf. Es wird bestimmt, wer welche Operationen auf eine Ressource anwenden darf. WebDAV führt dafür eigens den neuen Ressourcentyp **Principal** ein.

Principal

Ein Principal wird durch einen Bezeichner definiert. Dieser kann entweder ein Benutzer, eine Client Software, ein Server oder eine Gruppe mit weiteren Prinzipals sein.

Alle Operationen die an einer Ressource angewendet werden können, werden durch eine einzelne *Access Control List (ACL)* bestimmt. Diese ACL enthält wiederum *Access Control Entries (ACEs)*. Jede ACE definiert ein Principal und einen Satz von **Privilegien**, die durch die Principal entweder erlaubt (granted) oder verboten (denied) werden.

Privilegien

Privilegien repräsentieren eine oder mehrere HTTP Methoden und kontrollieren den Zugriff auf Ressourcen. Dabei werden die Privilegien in zwei Typen unterschieden:

- **Abstrakte Privilegien:**
  Abstrakte Privilegien können in einer ACE erlaubt (granted) oder verboten (denied) werden.
- **Aggregierte Privilegien:**
  Aggregierte Privilegien gruppieren zusammengehörige Privilegien. Zum Beispiel werden alle Privilegien für schreibende Vorgänge (write-acl, write-content, write-properties) zu einem globalen Schreib-Privileg zusammengefasst [Dus03].

Das folgende Beispiel zeigt, wie so ein Baum von Privilegien aussehen könnte:

```
[DAV:, all] (aggregate, abstract)
   |
   |
   +--- [DAV:, read] (aggregate)
                |
```

```
                |
                +--- [DAV:, read-acl] (abstract)
                |
                +--- [DAV:, read-current-user-privilege-set] (abstract)
    |
    |
    +--- [DAV:, write] (aggregate)
                |
                |
                +--- [DAV:, write-acl] (abstract)
                |
                +--- [DAV:, write-properties]
                |
                +--- [DAV:, write-content]
    |
    |
    +--- [DAV:, unlock]
.
```

Listing 2.13: Beispiel: ACL Struktur

Wird bei einem WebDAV Server beispielsweise ein aggregiertes write Privileg angeboten werden, kann bei der Serverimplementierung write-properties als abstraktes Privileg gesetzt werden. In diesem Fall kann das Privileg write-properties nicht direkt in einer ACE gesetzt werden. Sollte eine feinere Abstufung der Rechte erwünscht sein, werden write-content und write-properties als nicht-abstrakt gesetzt.

Folgende Privilegien regeln den Zugriff auf Ressourcen:

- **DAV:read**
  ist ein aggregiertes Privileg, um die Methoden zu steuern, die eine Ressource mit Metadaten liefert.
- **DAV:read-acl**
  Kontrolliert das Lesen einer ACL.
- **DAV:read-current-user-privilege-set**
  Kontrolliert die PROFIND Methode, um nur die Metadaten zu bekommen, die der momentane Benutzer bekommen kann. Dies ist sehr nützlich, um beispielsweise bei einer grafischen Oberfläche direkt die Aktionen auszublenden/auszugrauen die dem Benutzer sowieso nicht zur Verfügung stehen.
- **DAV:write**
  ist ein aggregiertes Privileg, um die Methoden zu Steuern, die eine Ressource oder deren Metadaten schreibt, verändert oder löscht.
- **DAV:write-properties**
  Privileg zum Steuern der Methoden, die Metadaten schreiben oder löschen.
- **DAV:write-content**
  Privileg zum Steuern der Methoden, die Ressourcen schreiben oder löschen.
- **DAV:write-acl**
  Privileg zum Steuern der Methode, die das Setzen einer ACL erlaubt.

- **DAV:bind**
  Steuert das Hinzufügen von Collections oder Ressourcen zu einer schon vorhandenen Collection.
- **DAV:unbind**
  Regelt das Löschen einer Ressource von einer Collection.
- **DAV:unlock**
  Regelt das Unlocken einer gelockten Ressource von einem fremden Benutzer, anders als dem Lock-Besitzer der immer den Lock entfernen darf.
- **DAV:all**
  aggregiertes Privileg, das alle anderen nicht-aggregierten Privilegien beinhaltet.

2.3.3.1 Access Control Properties

Mit der Einführung des Access Control Protocols, wird auch eine neue Anzahl von neuen Properties für WebDAV Ressourcen definiert. Die Access Control Properties werden durch die Request-URI identifiziert und können, wie auch bei anderen WebDAV Properties, durch die PROPFIND Methode erfragt werden.

HTTP-Ressourcen, die das Access Control Protocol unterstützen, müssen die folgenden Properties enthalten:

- **DAV:owner**
- **DAV:group**
- **DAV:supported-privilege-set**
- **DAV:current-user-privilege-set**
- **DAV:acl**
- **DAV:acl-restrictions**
- **DAV:inherited-acl-set**
- **DAV:principal-collection-set**

2.3.3.2 Betrachtung der Access Control List

Wie schon im vorangegangen Abschnitt zu erfahren war, ist die ACL ein Container für verschiedene Kontrollelemente, die den Zugriff auf Ressourcen regelt. Nachfolgend wird noch einmal die Struktur der ALC zusammengefasst.

- Eine ACL kann keine, ein oder mehrere Access Control Elemente (ACE) enthalten
- In der WebDAV-Hierarchie (Pfad zur Ressource) werden alle übergeordneten ACLs nach unten weiter vererbt
- An jedem WebDAV-Knoten kann eine oder keine ACL hängen

Es gibt eine gewisse Auflösungsregel für ACLs, abhängig von der Implementierung des jeweiligen Server. Durch die Weitervererbung nimmt die Gesamtanzahl der ACEs je ACL zwangsläufig nach unten hin zu. Wenn eine ACL einer Ressource modifiziert werden soll, muss immer die vollständige ACL gelesen, lokal geändert und dann wieder zurückgeschrieben werden.

Die folgende Vorgehensweise für einen Zugriff auf eine Ressource sieht wie folgt aus:

- Alle ACLs der Ressource (incl. aller geerbten) einlesen
- Alle Principals feststellen die zu dem augenblicklichen User gehören
- Alle ACEs aussortieren, die nichts mit dem User zu tun haben
- Alle ACEs in richtiger Reihenfolge übereinander legen
- Mit entsprechenden Algorithmus prüfen ob die geänderten Privilegien korrekt sind

Und so sieht dann eine ACE aus:

- Jede ACE enthält genau einen Principal
- Jede ACE kann nur eine oder mehrere Grant bzw. nur eine oder mehrere Deny Regeln enthalten
- Jede Grant/Deny Regel muß mindestens ein Privileg enthalten

Hierarchisiert sieht das dann wie folgt aus:

nichts	->	genau eins
“*”	->	beliebig viele
“+”	->	mindestens eins

```
ACL
|
|
+--- ACE*
       |
       |
       +--- Grant/Deny*
       |
       +--- Privilege+
.
```

Listing 2.14: Beispiel: ACE Struktur

Sollen für verschiedene Principals, Privilegien gesetzt werden, so muss für jedes Principal mindestens eine ACE gesetzt werden, weil beim gleichzeitigen Vorkommen von Grant und Deny zwei unabhängige ACEs gesetzt werden müssen (siehe Regel für Grant/Deny oben)[2].

2.4 WebDAV Applikationen

Es gibt eine ganze Reihe an Produkten und Werkzeugen, die das WebDAV Protokoll seit der Veröffentlichung unterstützen. Aufgrund des großen Umfangs verfügbarer Produkte ist eine vollständige Benennung aller Produkte nicht möglich und es wird sich auf die Benennung der vordergründigsten Produkte beschränkt. Es sei jedoch darauf hingewiesen, dass auf der Internetpräsenz von WebDAV (`www.webdav.org` [1]) eine aktuellere Liste der Produkte, die WebDAV unterstützen, vorgehalten wird.

2.4.1 Client

WebDAV Clients unterstützen den Benutzer, um sich zum WebDAV Servern zu verbinden und die Dateien und Verzeichnisse bearbeiten zu können.

- **DAVfs2**
  Ist ein Kommandozeilen-basierendes Programm für Unix. Es ist in Perl geschrieben und kann auch leicht für andere Plattformen eingesetzt werden [12].
- **Cadaver**
  Ist ein anderes Kommandozeilen basierter WebDav Client für Unix. Es unterstützt Upload, Download, namespace Operationen, (move/copy), Entwurf und Entfernung von Collections, locking Operationen [13].
- **Sitecopy**
  Realisiert autorenseitige Aktualisierung von Web- Servern. Es nutzt dabei wahlweise FTP oder WebDAV [14].
- **Microsoft Web Verzeichnisse und Office**
  Microsoft bindet die WebDAV Client Funktionalität in Windows und in

seinen Office Produkten ein. Ab Windows 2000 wird WebDAV direkt unterstützt. Es stellt eine Verbindung zu einem WebDAV Server über einen Assistenten her. Die Inhalte werden dann in einem virtuellen Verzeichnis als ein Teil desselben Dateisystems dargestellt [].

-> **Microsoft Office 2000**
Ermöglicht über alle Office 2000 Anwendungen das direkte Erstellen, Veröffentlichen, Bearbeiten und Speichern von Dokumenten in einem WebDAV Verzeichnis.

-> **Microsoft Internet Explorer ab 5**
Stellt die Verbindung zu einem WebDAV Verzeichnis her.

- **Dreamweaver**
Macromedia Dreamweaver ist ein beliebtes Webseiten-Entwicklungsprogramm.
- **Adobe Grafik und Web Entwicklungs- Produkte**
Adobe hat eine ganze Reihe an Produkten bei den die WebDAV Funktionalitäten implementiert wurden [9].

-> **Photoshop**

-> **GoLive 5**

-> **Acrobat**

- **Goliath**
Goliath wurde für Mac OS entwickelt. Es ist dafür gedacht, um Webseiten zu erstellen und zu editieren. Es war die erste Anwendung, die das WebDAV Protokoll für den Macintosh implementierte [15].
- **WebDrive**
WebDrive integriert einen WebDAV, HTTP oder FTP Server in den Windows-Desktop , indem er ihn als Laufwerk abbildet. Dadurch wird ermöglicht, dass der Benutzer mit jeder beliebigen Applikation auf die am Server gespeicherten Dateien zugreifen und diese ändern kann [16].
- **DAV Explorer**
Der DAV Explorer ist ein auf Java basierender WebDAV Browser mir vielen Erweiterungen [17].

2.4.2 Server

Auch die Anzahl der Webserver, die eine WebDAV Unterstützung vorweisen können, hat stark zugenommen.

- **Apache mit mod_dav Modul**
Apache ist das erste voll-kompatible Open-Source WebDAV Server. Das Modul unterstützt all Funktionalitäten die in RFC 2518 spezifiziert worden sind [18].

- **Jakarta Slide**
  Gehört mit zur Apache Software Foundation und ist ein Teil eines anderen Apache-Projekts. Jakarta Slide ist auch Open-Source und ein auf Java basierender WebDAV Server [19].
- **Zope**
  Ist ein weiterer Open-Source Web Application Server, implementiert in Python und C. Zope ist als Apache- CGI(ZAP) benutzbar [20].
- **Lighttpd**
  Lighttpd verfügt wie der Apache Server ein WebDAV unterstützendes Modul [21].
- **Internet Information Server (IIS)**
  Der Web-Server von Mircosoft Windows mir WebDAV Unterstützung [22].
- **Microsoft SharePoint**
- **Microsoft Exchange 2000**
- **Tamino WebDAV Server** Tamino ist ein Produkt der Software AG und eine Komponente des Tamino XML Servers [23].
- **Netware 5.1**
- **Oracle iFS (Internet File System)**
- **JigSaw (W3C HTTP- Referenzserver)**
- **Hyperwave Information Server 5.5**

Kapitel 3

Vorstellen der WebDAV Server Test Suites

3.1 Test Hintergrund

Um die Leistungsfähigkeit eines Server Systems feststellen zu können, müssen im Allgemeinen dazu verschiedene Messungen bzw. Tests durchgeführt werden. Ein wichtiges Indiz liefert hierzu deren Ressourcenverbrauch (CPU- und Speicher-Auslastung), die Verarbeitungsgeschwindigkeit und die Qualität der Ausgabe von Programmen und Hardware. Diese Art des Leistungsindikators wird in der Informatik auch als Performance bezeichnet.

Die Hauptaufgabe eines Servers ist vorrangig das Bereitstellen von Diensten für andere Systeme (Clients). Deshalb soll besonderes Augenmerk auf Messansätze gelegt werden, die die Ermittlung der Geschwindigkeit der Ausgabe von Servern zulassen.

Ein weiter wichtiger Punkt bei der Bewertung der Leistungsfähigkeit eines Servers ist die Konformität der implementierten Komponenten zu anderen Systemen. Die Güte der Zusammenarbeit zwischen zwei oder mehreren Systemen steigt mit der Konformität verwendeter Komponenten, da dadurch auch die Verarbeitungs- und Übertragungsfehler verringert werden, welche eine Zusammenarbeit eklatant stören können.

Die Leistungsfähigkeit eines Servers kann von der Serverseite, als auch von der Clientseite aus gemessen werden. Die serverseitigen Messungen liefern detaillierte Information über den Server, aber bringen einen höheren Messsystemaufwand mit sich. Ausserdem kann das Messprogramm, dass auf dem Server läuft, die Genauigkeit der Messungen stark beeinträchtigen. Im Gegensatz dazu führt die clientseitige Messung zu keinem weiteren Systemaufwand für den Server und reflektiert so das Leistungsverhalten der Server genauer. Das Problem bei diesem Ansatz ist, dass es zwischen dem Client und dem Server zu unterschiedlichen Netzlatenzzeiten kommen kann. Bei einer sorgfältigen Vor-

bereitung der Messumgebung und dem Prüfen der Interaktion zwischen Client und Server, kann die Genauigkeit dieses Messansatzes jedoch sehr verbessert werden.

3.2 Die Test Suites

In diesem Abschnitt des Kapitels, werden zwei Testprogramme (nachfolgend als *Test Suites* bezeichnet) vorgestellt. Mit deren Hilfe ist es möglich, die Leistungsfähigkeit - bezüglich der oben angesprochenen Performance und Konformität - eines Servers zu ermitteln. Beide Test Suites sind Open-Source Projekte und nutzten die *NEON WebDAV Bibliothek* [1] für die Kommunikation mit dem Server.

3.2.1 Litmus

Die Litmus WebDAV Server Protocol Compliance Test Suite ist, wie der Name schon andeutet, dazu ausgelegt, die Konformität des implementierten WebDAV Standards eines Servers zu ermitteln. Momentan gibt es zwei verschiedene Versionen von der Litmus Test Suite, wobei die erste Version die Konformität nach RFC 2518 (Standard WebDAV) und die zweite Version die Konformität nach RFC 3744 (Access Control Protocol) misst [4].

Es werden hierzu folgende Methoden für den Test nach RFC 2518 angewendet:

Namespace und Ressourcen Verwaltung

- **O**PTION mit DAV
  Feststellung und Überprüfung der vorhandenen WebDAV-Funktionalität und WebDAV-Versionierung
- **P**UT, GET
  Schreiben und Lesen von Daten mit anschließendem Bytevergleich
- **M**KCOL
  Neue Ressourcen des Typs Collection anlegen
- **D**ELETE
  Permanentes Entfernen einer Ressource oder einer Collection
- **C**OPY, MOVE
  Kopieren und Verschieben von Ressourcen in Kombination mit Überschreiben (Bestimmungsort besteht oder besteht nicht)

Property Verwaltung

- **P**ROPFIND, PROPPATCH
  Einstellen, Löschen und Ersetzen von Metadaten und Überprüfung, ob tote Metadaten nach COPY erhalten bleiben

Locking

- **L**OCK, UNLOCK
  Anwenden der LOCK Methoden auf abgeschlossene Ressourcen als Inhaber oder Nicht-Inhaber

Beim Test nach RFC 3744, werden hierzu folgende Methoden kombiniert angewendet:

Access Conrol

- **P**ROPFIND, PROPPATCH
  Einstellen, Löschen und Ersetzen von Prinzipien und Privilegien
- **A**CL
  Lesen und Modifizieren von ACLs und Erstellen von ACE
- **R**EPORT
  Beanspruchen eines erweiterten Mechanismus, um die Information einer Ressource zu lesen

Um zu gewährleisten, dass nach der Implementierung von WebDAV in einem Server die spätere Kommunikation mit einem WebDAV Client möglichst fehlerfrei abläuft, sollte beim Litmus-Test des Servers ein hohes Ergebnis erzielt werden.

Die nachfolgenden Ausschnitte eines Listings, zeigen exemplarisch die Ausgaben eines Testdurchlaufs
- nach RFC 2518 für einen Apache2 Server unter Linux
- und nach RFC 3744 für einen Jakarta Slide Server unter Linux

```
ubuntu@ubuntuserver: litmus http://192.168.27.130/webdav webdav webdav
-> running 'basic':
 0. init.................. pass
 1. begin................. pass
 2. options............... pass
 3. put_get............... pass
 4. put_get_utf8_segment.. pass
 5. mkcol_over_plain...... pass
 6. delete................ pass
 7. delete_null........... pass
 8. mkcol................. pass
 9. mkcol_again........... pass
........
........
........
........
24. owner_modify......... pass
25. double_sharedlock..... pass
26. notowner_modify...... pass
27. notowner_lock........ pass
28. unlock............... pass
29. finish............... pass
<- summary for 'locks': of 30 tests run: 30 passed, 0 failed. 100.0%
-> 1 warning was issued.
-> running 'http':
 0. init.................. pass
 1. begin................. pass
 2. expect100............. pass
 3. finish................ pass
```

```
<- summary for 'http': of 4 tests run: 4 passed, 0 failed. 100.0%
.
```

Listing 3.1: Beispiel: Litmus -Standard Test nach RFC 2518

```
root@ubuntuserver:/home/ubuntu/Desktop/litmus
./acl http://192.168.27.142:8080/slide/files root root
-> running 'acl':
 2. acl_init.............. pass
 3. acl_owner............. pass
........
........
........
........
13. acl_pcpl_prop_report.. pass
14. acl_pcpl_match_report. FAIL
(Error in REPORT acl-principal-prop-set)
15. aclget............... pass
16. acl_privileges_test... pass
17. acl_privileges_f..... pass
18. acl_copy_simple...... WARNING:
Resource COPY that should have failed. We dont have perms
........................ pass (with 1 warning)
19. aclcleanup........... pass
20. finish............... pass
<- summary for 'acl': of 21 tests run: 20 passed, 1 failed. 95.2%
-> 1 warning was issued.
root@ubuntuserver:/home/ubuntu/Desktop/litmus
.
```

Listing 3.2: Beispiel: Litmus - ACL Test nach RFC 3744

3.2.2 Prestan

Die Prestan WebDAV Server Performance Test Suite wurde dazu entwickelt, um die Performance hinsichtlich der Geschwindigkeit einzelner WebDAV Servermethoden zu ermitteln [5].

Prestan verwendet hierzu die WebDAV Methoden die auch nach RFC 2518 standardisiert wurden:

Namespace und Ressourcen Verwaltung

- **M**KCOL
  Collection und Sub-Collection anlegen
- **C**OPY
  Kopieren von einzelner/mehrerer Ressourcen und Collections
- **M**OVE
  Verschieben von einzelner/mehrerer Ressourcen und Collections
- **P**UT
  kleine/mittlere/große Ressourcen setzen
- **G**ET
  kleine/mittlere/große Ressourcen lesen

Property Verwaltung

- **P**ROPPATCH
  Properties einzelner/mehrerer Ressourcen setzen

- **P**ROPFIND
  Properties einzelner/mehrerer Ressourcen lesen

Locking

- **L**OCK, UNLOCK
  Anwenden der LOCK Methoden auf Ressourcen und Collections

Auch hier ein exemplarisches Listing für den Apache2 Server unter Linux

```
ubuntu@ubuntuserver:~/Desktop/Prestan
./Prestan http://192.168.27.130/webdav webdav webdav


Prestan, Version 2.0.3
Copyright(c) 2003 Teng Xu, GRASE Research Group at UCSC
http://www.soe.ucsc.edu/research/labs/grase

Server Warming Up.....Done

Start Testing http://192.168.27.130/webdav:


*********************************

* Number of Requests 10

* Number of Dead Properties 10

* Depth of Collection 10

* Width of Collection 100

* Type of Methods WebDAV

*********************************


ProppatchMult.................. Rsp = 1740 [us]
ProppatchSingle................ Rsp = 1670 [us]
PropfindDeadMult............... Rsp = 1549 [us]
PropfindDeadSingle............. Rsp = 1576 [us]
PropfindLiveMult............... Rsp = 2788 [us]
PropfindLiveSingle............. Rsp = 1624 [us]
Put1K.......................... Rsp = 1700 [us]
Get1K.......................... Rsp = 1506 [us]
Put64K......................... Rsp = 5261 [us]
Get64K......................... Rsp = 4941 [us]
Copy........................... Rsp = 2432 [us]
Move........................... Rsp = 2565 [us]
Delete......................... Rsp = 1779 [us]
MkCol.......................... Rsp = 1872 [us]
CopyCol........................ Rsp = 13547 [us]
MoveCol........................ Rsp = 11175 [us]
DeleteCol...................... Rsp = 8530 [us]
Lock........................... Rsp = 2193 [us
UnLock......................... Rsp = 1730 [us]
LockCol........................ Rsp = 7317 [us]
UnLockCol...................... Rsp = 5998 [us]
ubuntu@ubuntuserver:~/Desktop/Prestan
.
```

Listing 3.3: Beispiel: Prestan - Test

Kapitel 4

Konzeption und Implementierung des Access Control Protocols

4.1 Einleitung

Dieses Kapitel befasst sich mit der Konzeption und der Implementierung der ACP Erweiterung in den WebDAV Server Test Suites. Da die Litmus Test Suite, wie im Kapitel 3 zu erfahren war, in einer zweiten Version eine ACP Erweiterung erfahren hat, beschränkt sich die Konzeption und Implementierung nur auf die Prestan Test Suite.

Die Prestan Test Suite ist ein bestehendes Softwareprojekt. Aus diesem Grunde werden nachfolgend nur die grundlegendsten Phasen des Software Engineerings (Softwaretechnik) abgehandelt, an welchen sich im weiteren Verlaufe der Entwicklung orientiert werden soll, um einen strukturierten Entwicklungsprozess zu gewährleisten.

Bei der Entwicklung kommen die nachfolgend aufgeführten Phasen der Softwaretechnik zur Anwendung:

- die Analyse,
- der Entwurf,
- die Implementierung,
- und der Test der Implementierung

Diese Phasen stellen hier nur eine grobe Gliederung dar. In einigen Abschnitten kann es jedoch vereinzelt zu Überschneidungen zwischen den Phasen kommen.

4.2 Analyse

Die Analyse dient dazu, die genauen Aufgaben des eigentlichen Softwareprojekts vor Beginn zu klären. Dabei werden im Allgemeinen die Anforderungen und Ziele aus unterschiedlichen Blickwinkeln heraus betrachtet und dokumentiert.

Je nach Anwendungsfall des Softwareprojekts wird die Analyse in spezielle Teilgebiete untergliedert. Dadurch kann das Softwareprojekt in seine Bestandteile zerlegt und dann anschließend geordnet, untersucht und ausgewertet werden.

Speziell für die Prestan Test Suite kommen hier drei Teilgebiete der Analyse zum Einsatz:

- IST-Analyse
- Systemanalyse
- Anforderungsanalyse

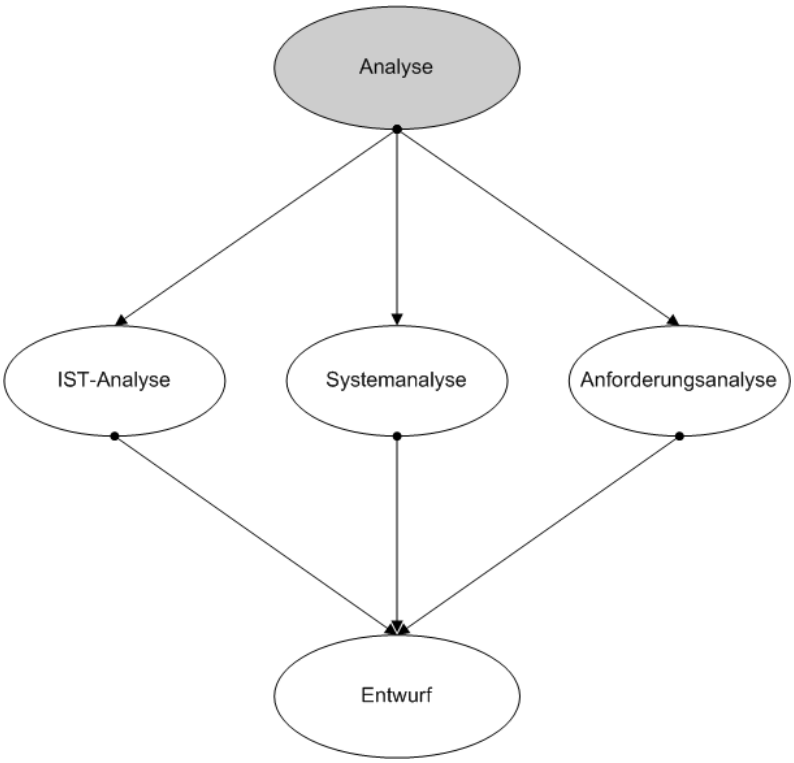

Abbildung 4.1: Struktur der Analyse Phase

4.2.1 IST-Analyse

IST-Analyse bedeutet, ein System in seinem gegenwärtigen Zustand zunächst aufzunehmen und zu dokumentieren. Die IST-Analyse wird dann eingesetzt, wenn das bereits existierende System erweitert oder ein System durch ein neues System der selben Art eingesetzt werden soll [Pre04].

Da die Prestan Test Suite ein bestehendes System (eine bestehende Software) ist, werden in einem ersten Schritt alle relevanten Daten, wie Bezeichner, Dokumentation, Source-Code, etc. betrachtet.

Die Ergebnisse der IST-Analyse sind der nachfolgenden Tabelle zu entnehmen:

Projekt Name	Prestan WebDAV Performance Test Suite
Version	0.2.0
Projekt-Art	Open-Source-Project
Code Standard	kein
Code Sprache	C
Daten Eingabe	Kommandozeile
Daten Ausgabe	Kommandozeile/Log-Datei
Dokumentation	keine
WebDAV Standards (implementiert)	RFC 2518
Source-Code Dateien	config.h common.h common.s basic.c props.c locks.c
Bibliotheken	NEON Lib
Bemerkung	—

Tabelle 4.1: Ergebnisse der IST-Analyse

4.2.2 Systemanalyse

Die *Systemanalyse* ist eine Methode, die auch bei bereits existierenden Systemen oder bei geplanten Systemen angewendet wird. Dabei werden alle relevanten Elemente und deren Beziehungen zueinander untersucht. Von dem zu untersuchenden System wird ein abstrahiertes Abbild als Modell einer sogenannten *Black-Box* geschaffen. Mit Hilfe des Black-Box-Tests lassen sich die Beziehungen zwischen der Daten-Eingabe (Input) und der Daten-Ausgabe (Output) darstellen.

Abbildung 4.2: Black-Box-Test - Systemanalyse

Angewendet auf die Prestan Test Suite konnten Ergebnisse gewonnen werden, welche in der folgenden Tabelle zusammengefasst sind.

Input nur auf der Kommandozeile	
Hilfeaufruf	Prestan –help
Programmaufruf	Prestan bzw. ./Prestan
Parameter und Optionen für die URI	Protokoll (http) Zieladresse Port Pfad user password -r Anzahl der Requests -p Anzahl der Dead Properties -d Tiefe der Struktur von Collection -w Anzahl der Collection nach dem erreichen der Tiefe -o Name der Ausgabedatei -m Typ der WebDAV Methode

Tabelle 4.2: Black-Box-Test - Systemanalyse - Ergebnisse Input

Output auf der Kommandozeile oder in einer Logdatei	
Prestan Hilfe	Usage: ./Prestan [http://]hostname[:port]/path + [username password] [options] Option: -r, - -Requests Number of repeat requests (Default: 10) -p, - -Properties Number of dead properties (Default: 10) -d, - -Depth Depth of collection (Default: 10) -w, - -Width Width of collection at the bottom level (Default: 100) -o, - -Output Output file -m, - -Methods Type of Web Methods (WebDAV/WebFolder, Default: WebDAV) Example: ./Prestan http://dav.cse.ucsc.edu:81/basic test1 test1 -r 20 -p 20 -m WebFolder

Tabelle 4.3: Black-Box-Test - Systemanalyse - Ergebnisse Output1

Output auf der Kommandozeile oder in einer Logdatei	
Methoden und Ergebnisse	ProppatchMult ProppatchSingle PropfindDeadMult PropfindDeadSingle PropfindLiveMult PropfindLiveSingle Put1K Get1K Put64K Get64K Copy Move Delete MkCol CopyCol MoveCol DeleteCol Lock] UnLock LockCol UnLockCol

Tabelle 4.4: Black-Box-Test - Systemanalyse - Ergebnisse Output2

4.2.3 Anforderungsanalyse

Ein weitere Methode der Software-Analyse ist die Anforderungsanalyse. Wie der Name schon sagt, soll die Anforderungsanalyse die Anforderungen reflektieren, die für die Modellierung des Ziel-Systems benötigt werden. Dabei fließen die Analyseergebnisse der IST-Analyse und der Systemanalyse mit ein [And02].

Die Definition einer Anforderung (Requirement) lautet nach IEEE[24] 610.121990 wie folgt:

"Die Anforderung ist eine Bedingung oder Fähigkeit, die eine Software erfüllen oder besitzen muss, um einen Vertrag, eine Norm oder ein anderes, formell bestimmtes Dokument zu erfüllen [25]."

Werden nun alle Ergebnisse der Analyse zusammengefasst, können für die Prestan Test Suite, folgende Anforderungen definiert werden:

- Anpassen und Erweitern der Standard Test-Methoden an die Anforderungen des DLRs,
- für eine bessere Auswertung der Ergebnisse, sollte die Ausgabe mit Hilfe von XML erweitert werden,
- Implementieren der ACP Erweiterung

Anpassung und Erweiterung der vorhandenen Test-Methoden

Die Analyse hat gezeigt, dass die Prestan Test Suite nur Dateien (Ressourcen) bis zu einer Gesamtgröße von 64KB testet. Da aber die Größe von einigen wissenschaftlichen Dokumenten oder auch anderen Dateien größer als 1MB sein können, entspricht dies nicht mehr den heutigen Anforderungen des DLRs für WebDAV Server.

Um diesen Anforderungen des DLRs zu genügen, muss getestet werden, wie sich ein WebDAV Server auch mit Dateien von 1MB bis 5MB Größe verhält. Dazu werden die im Prestan Quellcode befindlichen Test-Methoden, die für den Aufruf und der Übertragung von Dateien zuständig sind, erweitert.

Der nachfolgende Ausschnitt des Quellcodes, zeigt die oben erwähnten Test-Methoden.

```
static int do_put_get(const char *segment, int fsize)
{
    ......
    ......

    /* PUT METHOD */
    SEND_REQUEST(ne_put(i_session, uri, fd));
    memset(str, 0, sizeof(str));
    sprintf(str, "Put%dK", fsize);

    ......
    ......
```

```
    /* GET METHOD*/
    fd = mkstemp(tmp);
    BINARYMODE(fd);

    SEND_REQUEST(ne_get(i_session, uri, fd));
    memset(str, 0, sizeof(str));
    sprintf(str, "Get%dK", fsize);

    ......
    ......
}


int put_get1K(void)
{
    return do_put_get("res", 1);
}


int put_get64K(void)
{
    return do_put_get("res", 64);
}
```

Listing 4.1: Prestan: PUT_GET Test-Methoden

XML-Ausgabe

Die Analyse hat weiterhin gezeigt, dass die Prestan Test Suite, alternativ zur Kommandozeilenausgabe, nur die Logdateiausgabe zu Verfügung stellt. Dies reicht aber für eine erweiterte Auswertung der Ergebnisse bei Weitem nicht aus.

Durch die Erweiterung mit einer XML-Ausgabe lassen sich die Ergebnisse durch andere bereits existierende Programme besser bearbeiten. Weiterhin besteht dann auch die Möglichkeit, nach bestimmten Elementen zu suchen oder die Ergebnisse grafisch darzustellen.

ACP Erweiterung

Die Funktionalitäten des Access Control Protocols sind im Kapitel 2 und im RFC 3744 ausführlich beschrieben worden. Deshalb wird sich in der nachfolgenden Entwurfsphase darauf bezogen.

4.3 Entwurf

In der Entwurfsphase findet die eigentliche Modellierung des Ziel-Systems statt. Mit Hilfe von verschiedenen Techniken der Softwareentwicklung werden aus den Anforderungen abstrakte Modelle geschaffen.

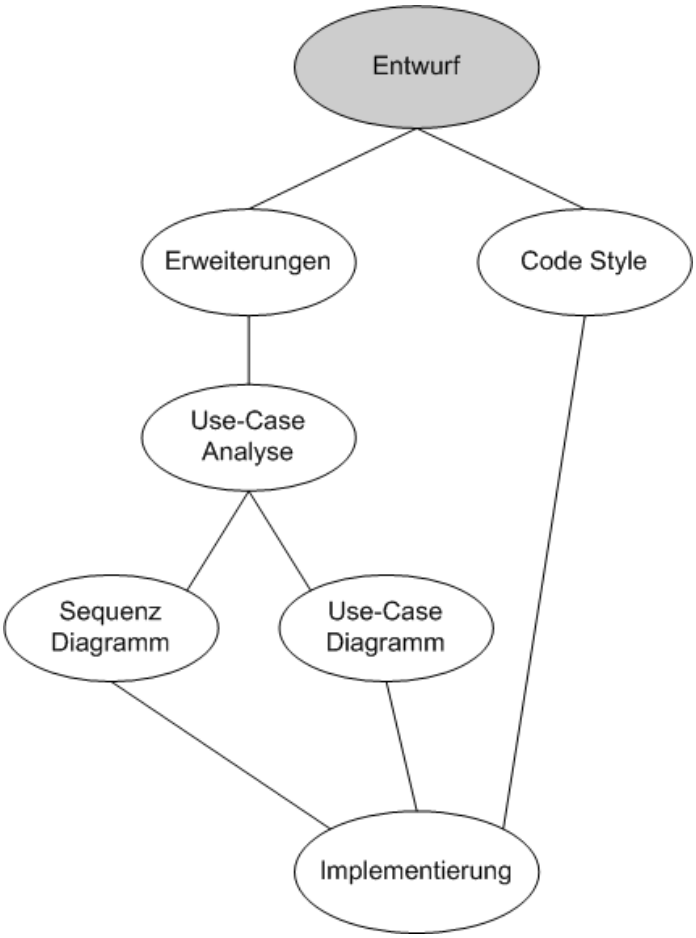

Abbildung 4.3: Struktur der Entwurfs Phase

4.3.1 Use-Case Analyse

Die Use-Case Analyse ist eng mit Analyse verknüpft und stellt eine Methode dar, mit der die relevanten Anforderungen des Zielsystems in entsprechende Anwendungsfälle (Use Cases) transformiert und in einem Use-Case Diagramm visualisiert werden können [Qua98].

Angewendet auf die Prestan Test Suite werden hier nachfolgend folgende Anwendungsfälle definiert, die in drei Gruppen eingeteilt werden können.

- Beginn des Tests: Gruppe UC000
- Lesen und Schreiben von ACL Elementen an verschiedenen Ressourcen-Typen: Gruppe UC100
- Beenden des Tests mit Übermittlung der Ergebnisse in dem vorher festgelegten Ausgabeformat: Gruppe UC200

Beginn des Tests

- **UC001: Initiierung des Performance Test ->** Hiemit wird der eigentliche Test durch den Benutzer gestartet. Dabei müssen vorher alle nötigen Zielparameter des zu testenden Servers bekannt sein.

Anwendungsfall	**UC001: initiieren des Performance Tests**
Aktoren	User
Auslöser	User
Vorbedingungen	- Verbindung zum Server ist vorhanden - Ziel- und Testparameter sind bekannt
Verlauf **Verlauf**	- Parameter eingeben - Test mit Enter starten

Tabelle 4.5: Use-Case UC001 Initiierung des Performance Test

Lesen und Schreiben von ACL Elementen

- **UC101: get DAV:ACL Single Ressource ->** Da eine einfache Ressource als Grundtyp angesehen werden kann, ist das Auslesen der ACL Elemente einer einzelnen Ressource auch der erste Anwendungsfall dieser Gruppe.

Anwendungsfall	**UC101: get DAV:ACL Single Ressource**
Aktoren	User
Auslöser	Prestan Test Suite (Client)
Vorbedingungen	- eine Ressouce wurden angelegt
Verlauf	- der Client sendet ein Request um Informationen über die ACL einer einzelnen Ressource zu erhalten - Server stellt die ACL mit den einzelnen ACEs zusammen - Server sendet ein Response mit der ACL

Tabelle 4.6: Use-Case UC101 get DAV:ACL Single Ressource

- **UC102: set DAV:ACL Single Ressource ->** Analog zum Auslesen der ACL Elemente verhält sich das Bescheiben einer einzelnen Ressource.

Anwendungsfall	**UC102: set DAV:ACL Single Ressource**
Aktoren	User
Auslöser	Prestan Test Suite (Client)
Vorbedingungen	—
Verlauf	- der Client sendet ein Request mit einer ACL für die einzelne Ressource - Server liest die ACL der Ressource aus - Server ändert die ACL mit den Modifikationen der empfangenen ACL - Server schreibt die geänderte ACL auf die Ressuorce - Server sendet ein Response mit der Statusmeldung

Tabelle 4.7: Use-Case UC102 set DAV:ACL Single Ressource

- **UC103: get DAV:ACL Single Collection ->** Die Collection ist zwar ein weiterer Ressorcentyp, aber prinzipell unterscheiden sich das Auslesen der ACL Elemente kaum von der einer einfachen Ressource. Von Bedeutung wäre hier aber: wie behandelt der Server eine einzelne Collection unter der Betrachtung der Speicherung, Index, etc.?

Anwendungsfall	**UC103: get DAV:ACL Single Collection**
Aktoren	User
Auslöser	Prestan Test Suite (Client)
Vorbedingungen	- ein Collection wurden angelegt
Verlauf	- der Client sendet ein Request um Informationen über die ACL der einzelnen Collection zu - Server stellt die ACL mit den einzelnen ACEs zusammen - Server sendet ein Response mit der ACL

Tabelle 4.8: Use-Case UC103 get DAV:ACL Single Collection

- **UC104: set DAV:ACL Single Collection ->** Auch hier analog zu UC103: das Beschreiben der einzelnen Collection ohne weitere Kind-Ressourcen (Collection, Ressourcen).

Anwendungsfall	**UC104: set DAV:ACL Single Collection**
Aktoren	User
Auslöser	Prestan Test Suite (Client)
Vorbedingungen	—
Verlauf	- der Client sendet ein Request mit einer ACL für die einzelne Collections - Server liest die ACL der Collection aus - Server ändert die ACL mit den Modifikationen der empfangenen ACL - Server schreibt die geänderte ACL auf die Collection - Server sendet ein Response mit der Statusmeldung

Tabelle 4.9: Use-Case UC104 set DAV:ACL Single Collection

- **UC105: get DAV:ACL Multi Collection ->** Dieser Anwendugsfall betrachtet das Auslesen der ACL Elemente an einer Collection, die noch weiter Kind-Ressourcen beinhaltet.

Anwendungsfall	**UC105: get DAV:ACL Multi Collection**
Aktoren	User
Auslöser	Prestan Test Suite (Client)
Vorbedingungen	ein Collection mit mehreren Sub-Ressourcen wurden angelegt
Verlauf	- der Client sendet ein Request, um Informationen über die ACL der (Vater)Collection zu bekommen - Server stellt die ACL mit den einzelnen ACEs zusammen - Server sendet ein Response mit der ACL

Tabelle 4.10: Use-Case UC105 get DAV:ACL Multi Collection

- **UC106: set DAV:ACL Multi Collection ->** Eine Collection vererbt alle Eigenschaften seiner ACL Elemente an die der Kind-Ressourcen.

Anwendungsfall	**UC106: set DAV:ACL Multi Collection**
Aktoren	User
Auslöser	Prestan Test Suite (Client)
Vorbedingungen	—
Verlauf	- der Client sendet ein Request mit einer ACL für eine Collection - Server lisßt sequentiell oder parallel die ACL der Collection und Sub-Ressourcen (Collection und Ressourcen) aus - Server ändert sequentiell oder parallel die ACL mit den Modifikationen der empfangenen ACL - Server schreibt sequentiell oder parallel die geänderte ACL auf jeweilige Ressource - Server sendet ein Response mit der Statusmeldung

Tabelle 4.11: Use-Case UC106 set DAV:ACL Multi Collection

Beenden des Tests

- **UC201: ausgeben der Ergebnisse als XML-File ->** Alle gesammelten Ergebnisse werden in einer XML-Datei geschrieben und anschließend wird der Test beendet.

Anwendungsfall	**UC201: create XML-File**
Aktoren	User
Auslöser	Prestan Test Suite (Client)
Vorbedingungen	- Test verlief ohne Fehler
Verlauf	- anlegen und öffnen einer XML-Datei - schreiben der Ergebnisse in die XML-Datei - schließen der XML-Datei

Tabelle 4.12: Use-Case UC201 XML Ergebnis Ausgabe

4.3.2 Use-Case Diagramm

Die aus der Use-Case Analyse erstellten Anwendungsfälle werden im Use-Case Diagramm, als Beziehung im gesamten System zueinander dargestellt.

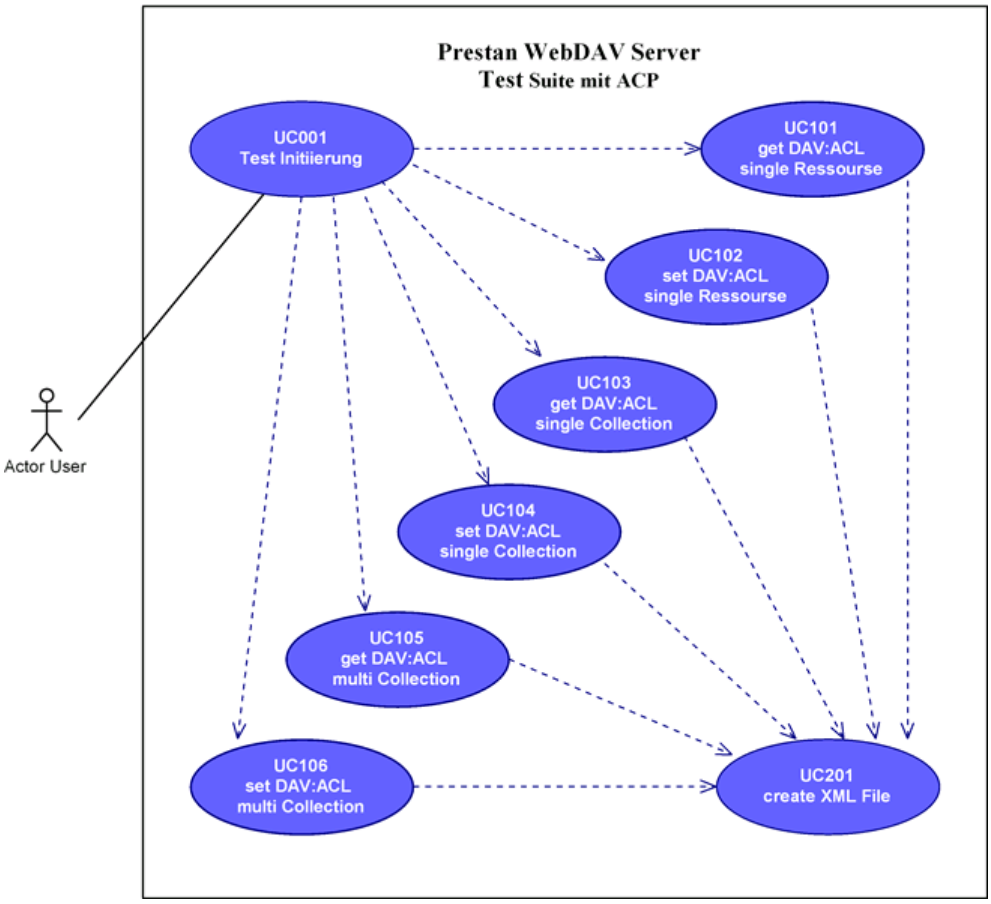

Abbildung 4.4: Prestan: Use-Case-Diagramm

4.3.3 Sequenz-Diagramm

Das Sequenz-Diagramm zeigt die Interaktion zwischen den beteiligten Systemkomponenten auf, so dass die zeitliche Sequenz der Nachrichten und die damit verbundenen Ereignisse nachverfolgt werden können.

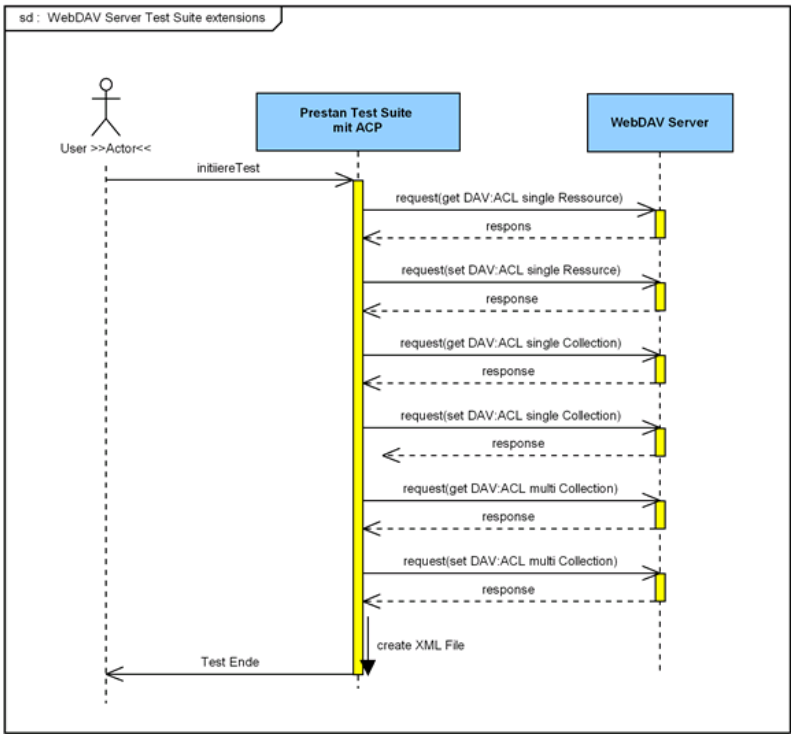

Abbildung 4.5: Prestan Sequenz-Diagramm

4.3.4 Code-Style

Wichtig für ein Softwareprojekt ist auch das Festlegen des *Code-Styles*. Der Code Style sollte sich allgemein immer nach Regel eines so genannten "Coding Standards" richten. Diese Regel wurden vor allem von Firmen, Universitäten oder andere Organisationen aufgestellt, um die Zusammenarbeit zwischen vielen Programmierern handhabbar zu machen.

Ziele dieser Standards sind zum Beispiel:

- die Lesbarkeit des Programmcodes durch sinnvolle Namensgebungen und einheitliche Strukturierung zu erhöhen,
- die Portabilität auf andere Rechner oder Plattformen als die Entwicklungsplattform sicherzustellen, sowie
- die Erweiterbarkeit und Wiederbenutzbarkeit der Programme,
- und die Fehlerfreiheit zu erreichen.

Da die Prestan WebDAV Server Test Suite ein Open-Source Projekt ist und demzufolge viele Entwickler an diesem Projekt arbeiten können, ist es um so wichtiger, dass der Programmcode einer vorgegebenen Richtlinie folgt. Aber aus der eingeschränkten Dokumentation ist keine solche Richtlinie oder Ähnliches zu entnehmen. Auch durch die Analyse des Programmcodes, konnte nicht eindeutig festgestellt werden, nach welchen "Coding Standard" hier gearbeitet wurde. Deshalb werden alle nachfolgende Erweiterungen in dem gängigen GNU-Style programmiert.

Klammersetzung

Für die Klammersetzung besagt der GNU-Style [Ric07], dass sich die zusammengehörigen Klammerpaare in einer Spalte zu befinden haben.

```
int main()
{
   int i;

   for (i=0; i<=value; i++)
   {
   ........
   ........
   }

}
```

Listing 4.2: Beispiel: Klammerpaare

Variablennamen

Variablennamen werden komplett klein geschrieben und die Wörter knapp aber sinngemäß durch den Unterstrich getrennt. Ausnahmen sind Konstanten, Makros und Aufzählungsvariablen die durch eine durchgängige Großschreibung angezeigt werden.

Beispiel:

row_counter ist richtig, aber *rowcounter* und *rowCounter* sind es nicht

Wichtiger als solche formalen Präfix-Regeln ist allerdings die Wahl aussagekräftiger und prägnanter Namen.

Variablendeklaration

Pro Zeile darf nur eine Deklaration stattfinden. Diese Form eignet sich besser zur Dokumentation und vermeidet Fehler wie:

```
{
int* a, b, c; // a ist pointer auf int, b und c sind int!

    ........
    ........

}
```

Listing 4.3: Beispiel: Variablendeklaration

Variablen werden möglichst lokal deklariert und an der Stelle ihrer Initialisierung.

Funktionsnamen und Methoden

Mit Funktionsnamen verhält es sich ähnlich wie mit Variablennamen. Sie sollten sinnvoll und beschreibend sein. Hier gilt auch die Kleinschreibung und Unterstriche als Trennung zwischen den Wörtern. Funktionen sollten ein Verb im Namen haben.

Gute Funktionsnamen sind zum Beispiel:

```
int print_login_status()
{
    ........
    ........
}

int get_user_data()
{
    ........
    ........
}
```

Listing 4.4: Beispiel: Funktionsnamen

Funktionsparameter

Die Funktionsparameter richten sich nach denselben Regeln wie die der Variablennamen. Es sollten keine Parameter wie do_stuff(int a, int b, char c) sein.

Bei vielen Funktionsargumenten wird die Argumentenliste entsprechend umgebrochen und in der nächsten Zeile fortgesetzt.

Kommentare

Kommentare spielen in der Programmierung eine wichtige Rolle, die aber leider von einigen Programmierern sehr vernachlässigt wird. Jeder komplexen Funktion sollte ein Kommentar vorhergehen die dem Programmierer alles erklärt was er braucht, um zu wissen, wie die Funktion funktioniert. Die Bedeutung jedes Parameters, die erwarteten Werte, und die Rückgabe Werte sind minimale Voraussetzung.

Kommentare werden von einigen Programmierhilfen (javadoc, kdoc, doxygen) dazu verwendet, automatisch Dokumentationen zu erzeugen. Dazu ist jeweils eine spezielle Syntax der Kommentare zu verwenden.

Bei der Prestan Test Suite sind die Funktionen zum größten Teil schlecht oder gar nicht kommentiert. So ist auch eine Generierung einer vollständigen Code-Dokumentation zurzeit nicht möglich. Der Code der Erweiterungen wird möglichst ausführlich kommentiert und nach folgenden Format gegliedert:

- Am Anfang jeder Datei ein ausführlicher und sinnvoller Kommentar, der die programmierte Funktionalität beschreibt
- Vor jeder Funktion eine Beschreibung, was die Funktion macht
- Kommentare werden “allgemein” formuliert sein
- Kommentare in der gleichen Sprache wie Variablennamen
- Kommentare kommen möglichst vollständig in den Kopf einer Funktion [Ric07].

Formatierung

Um eine gute Lesbarkeit des Programmtextes mit gängigen Texteditoren zu gewährleisten, werden hier nachfolgend ein paar Regeln aufgeführt.

- Es werden nicht mehr als 80 Zeichen pro Zeile verwenden.
- Die Tabs werden durch 4 Leerzeichen ersetzt.
- Geschweifte Klammern werden konsistent gesetzt werden und paarweise untereinander stehen.
- Funktionen werden kurz gehalten und wenn nötig durch die Einführung weiterer Funktionen gekürzt, so dass der Inhalt einer Funktion idealerweise immer auf einen Bildschirm passt.

4.4 Implementierung

In der Implementierung Phase werden nun alle abstrakten Modelle der Entwurfphase konkret umgesetzt.

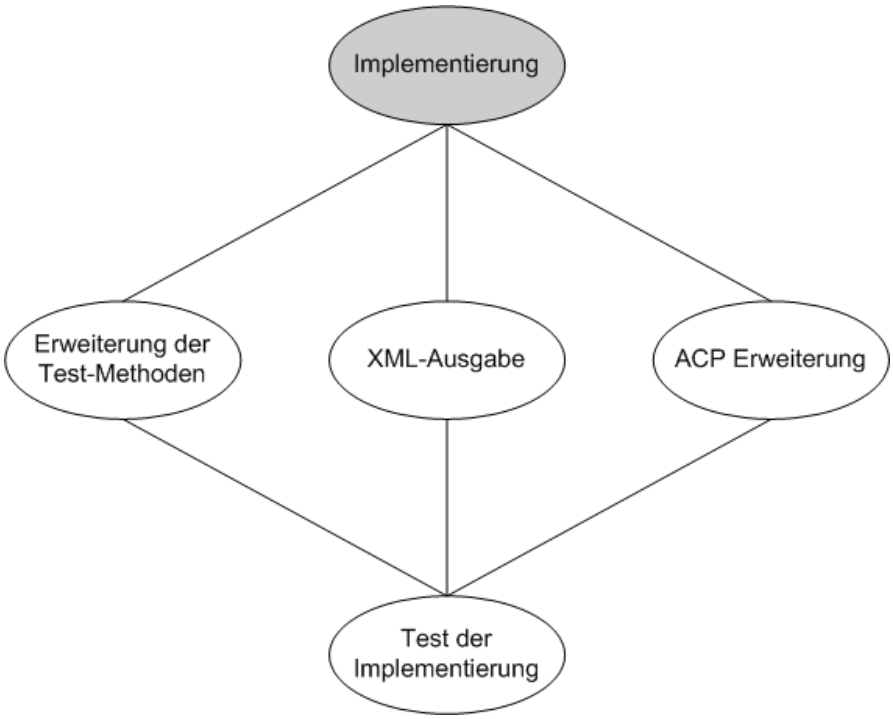

Abbildung 4.6: Struktur der Implementierungs Phase

4.4.1 Implementierung der erweiterten Test-Methoden in Prestan

Die Implementierung der erweiterten Test-Methoden war recht trivial. Da die Struktur und Funktionalität der Test-Methoden schon gegeben waren, mussten diesen nur übernommen und mit den entsprechenden Parametern angepasst werden.

Es wurden drei Modifikationen der vorhandenen Dateien vorgenommen:

1. **Hinzufügen von weiteren Funktionen in der Datei basic.c:**

```
int put_get1024K(void)
{
   return do_put_get("res", 1024);
}

int put_get2048K(void)
{
   return do_put_get("res", 2048);
}

int put_get3072K(void)
{
   return do_put_get("res", 3072);
}

int put_get4096K(void)
{
   return do_put_get("res", 4096);
}

int put_get5120K(void)
{
   return do_put_get("res", 5120);
}
```

Listing 4.5: Prestan: Erweiterung der PUT_GET Methoden

2. **Anpassen der Header Datei common.h:**

```
int put_get1024K(void);
int put_get2048K(void);
int put_get3072K(void);
int put_get4096K(void);
int put_get5120K(void);
```

Listing 4.6: Prestan: Anpassung der Header Datei

3. **Erweitern der Test-Matrix in der common.c Datei:**

```
ne_test tests[] =
{
   INIT_TESTS,

   T(propinit),
   T(proppatch),
   T(propfinddead),
   T(propfindlive),

   T(put_get1K),
   T(put_get64K),
   T(put_get1024K)         //new,
   T(put_get2048K)         //new,
   T(put_get3072K)         //new,
   T(put_get4096K)         //new,
   T(put_get5120K)         //new,
```

```
    T(my_single),
    T(my_collection),

    T(locks),

    T(acl)                  //new,

    FINISH_TESTS
};
```

Listing 4.7: Prestan: Erweiterung der Test-Matrix

4.4.2 Implementierung der XML-Ausgabe in Prestan

Für die Implementierung der XML-Ausgabe wurde eigens eine neue Datei mit neuen Funktionen hinzugefügt. Es wurde absichtlich auf die Implementierung andere XML-Parser verzichtet. Gründe hierfür waren zum einen, dass die meisten XML-Parser sehr allgemein gehalten und überladen sind. Auf diese Weise konnte der Programmcode übersichtlicher gehalten werden. Zum anderen wurde für die XML-Ausgabe das Format einer Excel XML-Datei gewählt. Damit können die Ergebnisse sehr gut in Excel grafisch aufbereitet und in Form von Balkendiagrammen dargestellt werden.

Unter Berücksichtigung der vorangegangenen Anmerkungen sind folgende Funktionen entstanden:

1. **Erzeugen und Öffnen einer XML-Datei:**
   Die Funktion open_xml_file(void) legt mit fopen() eine XML-Datei an und beschreibt (fprindf(...)) diese mit dem vorbereiteten Excel XML-Header.

```
int open_xml_file(void)
{
    ........
    ........

    f = fopen(tfile , "w+");

    if (f)
    {
        printf("\nopen XML-File : %s \n\n", tfile);

        fprintf(f,"<?xml version=\"1.0\"?>\n");
        fprintf(f,"<Workbook");
        fprintf(f,"xmlns=\"urn:schemas-microsoft-com:office:spreadsheet\"\n");
        fprintf(f,"xmlns:x=\"urn:schemas-microsoft-com:office:excel\"\n");
        fprintf(f,"xmlns:ss=\"urn:schemas-microsoft-com:office:spreadsheet\">");
        fprintf(f,"\n <Worksheet ss:Name=\"Table1\">\n");
        fprintf(f,"  <Table>\n\n");

        return 0;
    }
    else printf("error open XML-File!\n");

    return 1;
}
```

Listing 4.8: Prestan: öffnen der XML-Datei

2. **Schreiben der Daten in die XML-Datei:**
   Die drei folgenden Funktionen bekommen die Ergebnisparameter von den

Prestan Test-Methoden und schreiben die Daten je nach Typ und Anzahl als Zelle in die XML-Datei.

```
int write_row_cell_s_xml_file(char *string, int rowindex)
{
   fprintf(f,"<Row ss:Index=\"%i\">\n", rowindex);
   fprintf(f," <Cell><Data ss:Type=\"String\">%s</Data></Cell>\n", string);
   fprintf(f,"</Row>\n");

   return 0;
}


int write_row_2cell_sn_xml_file(char *string, int number, int rowindex)
{
   fprintf(f,"<Row ss:Index=\"%i\">\n", rowindex);
   fprintf(f," <Cell><Data ss:Type=\"String\">%s</Data></Cell>\n", string);
   fprintf(f," <Cell><Data ss:Type=\"Number\">%i</Data></Cell>\n", number);
   fprintf(f,"</Row>\n");

   return 0;
}


int write_row_2cell_ss_xml_file(char *string1, char *string2, int rowindex)
{
   fprintf(f,"<Row ss:Index=\"%i\">\n", rowindex);
   fprintf(f," <Cell><Data ss:Type=\"String\">%s</Data></Cell>\n", string1);
   fprintf(f," <Cell><Data ss:Type=\"String\">%s</Data></Cell>\n", string2);
   fprintf(f,"</Row>\n");

   return 0;
}
```

Listing 4.9: Prestan: schreiben in XML-Datei

3. **Schließen der XML-Datei:**
   Die Funktion close_xml_file(void) schließt die XML-Datei, nachdem alle Prestan Tests abgeschlossen wurden und der XML-Foot geschrieben wurde.

```
int close_xml_file(void)
{
   fprintf(f,"\n  </Table>\n");
   fprintf(f," </Worksheet>\n");
   fprintf(f,"</Workbook>\n");

   if (!fclose(f))
   {
       printf("\nclose XML-File : %s \n\n", tfile);

       return 0;
   }
   else printf("error close XML-File!\n");

   return 1;
}
```

Listing 4.10: Prestan: schließen der XML-Datei

4.4.3 Implementierung der ACP Erweiterung in Prestan

Bei der Implementierung der ACP Erweiterung wurde, wie bei den anderen WebDAV Erweiterungen, eine eigene Datei mit neuen Funktionen hinzugefügt. Außerdem wurde darauf geachtet, dass der strukturelle Aufbau der Datei mit den Funktionen und Methoden, dem der anderen Dateien gleicht.

Folgende Funktionen wurden für die ACP Erweiterung entwickelt:

1. **Start Methode:** Jede Datei einer WebDAV Erweiterung besitzt eine sogenannte Start-Methode. Diese wird von der Test-Matrix in der Hauptdatei aufgerufen. Diese Start Methode, ruft wiederum die eigentlichen Funktionen, auf die für den Test notwendig sind.

```
int acl(void)
{
   acl_init();
   send_request(acl_uri, "OPTIONS", "");
   send_request(acl_uri, "ACL", "");

   get_acl(acl_single_res_uri, "GetAclSingleRes");
   set_acl(acl_single_res_uri, "SetAclSingleRes");

   get_acl(acl_single_coll_uri, "GetAclSingleColl");
   set_acl(acl_single_coll_uri, "SetAclSingleColl");

   get_acl(acl_multi_coll_uri, "GetAclMultiColl");
   set_acl(acl_multi_coll_uri, "SetAclMultiColl");

   send_request(acl_uri, "DELETE", "");

   return 0;
}
```

Listing 4.11: Prestan: ACP Erweiterung - beginne ACL Test

2. **acl_init Funktion**
   Unter Zuhilfename von Funktionen aus der NEON-Bibliothek, erzeugt die acl_init() Funktion verschiedene Ressourcen, an denen die ACL Tests durchgeführt werden. Weiterhin werden die URIs der jeweiligen Ressourcen gespeichert.

```
int acl_init(void)
{
   // Create base collection called acl-test
   acl_uri=ne_concat(i_path,"acl-test/",NULL);
   ONNREQ("could not create collection", ne_mkcol(i_session, acl_uri));


   acl_single_res_uri = ne_concat(i_path, "acl-test/aclsingleres", NULL);
   CALL(upload_foo("acl-test/aclsingleres"));

   ........
   ........

   return 0;
}
```

Listing 4.12: Prestan: ACP Erweiterung - Anlegen der zu testenden Ressourcen

3. **get_acl und set_acl Funktionen**
   Die Funktionen get_acl und set_acl sind für das Lesen und Beschreiben von ACLs einer Ressource zuständig. Mit dem Aufruf einer dieser Funktionen, wird die entsprechende Struktur des ACL-Requests zusammengestellt. Der so erzeugte ACL-Request, wird der Funktion übergeben, die für den eigentlichen Versand der Request zuständig ist. Gleichzeitig wird die Antwortzeit des Servers auf diesen Request gemessen.

```
int get_acl(const char *uri, char *msg)
{
    PRECOND(acl_ok);

    SEND_REQUEST(send_request(uri, "PROPFIND", acl_get_body));

    my_printf(msg);
    return 0;
}


int set_acl(const char *uri, char *msg)
{
    PRECOND(acl_ok);

    SEND_REQUEST(send_request(uri, "ACL", acl_set_body));

    my_printf(msg);
    return 0;
}
```

Listing 4.13: Prestan: ACP Erweiterung - Lesen und Beschreiben von ACLs

4. **send_request Funktion**
   Die send_request Funktion übernimmt die Struktur der ACL Requests und sendet diese mit Hilfe der Funktionen aus der NEON-Bibliothek an den Server.

```
int send_request(const char *uri, char *request, char *request_body)
{
    ne_request *req = ne_request_create(i_session, request, uri);
    ne_add_depth_header(req, NE_DEPTH_ZERO);
    ne_set_request_body_buffer(req, request_body, strlen(request_body));

    ........
    ........

    ne_request_destroy(req);
    return 0;
}
```

Listing 4.14: Prestan: ACP Erweiterung - Senden der Requests

4.5 Softwaretest

Nach dem die Implementierungen abgeschlossen wurde, gilt es nun die Funktionalität der Erweiterungen zu testen. Dazu werden für Prestan drei Methoden angewendet.

- Unit-Test: Testen von einzelnen Komponenten
- Black-Box-Test: Vergleich zwischen Input und Output
- Netzwerk Protokoll Analyse: Analysieren des Datenverkehrs zwischen Client (Prestan) und Server

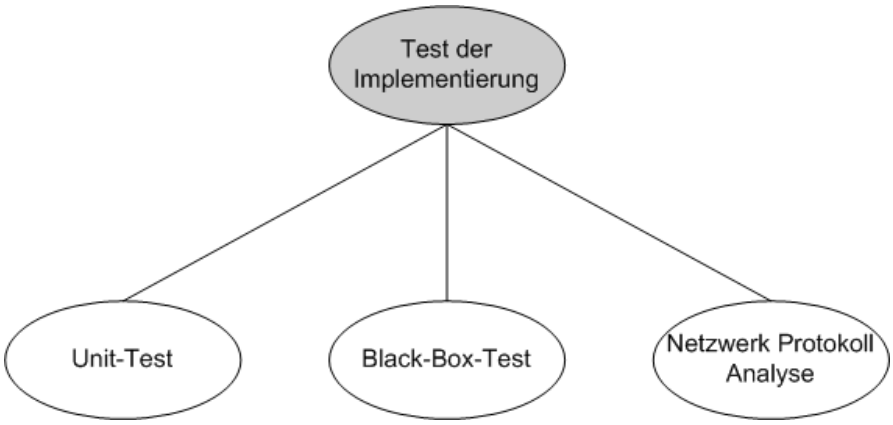

Abbildung 4.7: Struktur der Softwaretest Phase

4.5.1 Unit-Test

Der *Unit-Test* (auch Komponententest oder Modultest) dient zur Überprüfung der Korrektheit von Modulen einer Software, beispielsweise von Funktionen und Methoden. Bei der Programmierung solch eines Modules kann es zu verschiedenartigen Fehlern kommen. Um eben jene Fehler zu finden, muss der Programmcode getestet werden.

An dieser Stelle setzen die so genannten *Unit-Tests* ein. Dabei werden bei jedem Test die zu testende Funktion oder Methode mit Testdaten (Parametern) konfrontiert und deren Reaktion auf diese Testdaten geprüft. Die zu erwartenden Ausgabewerte werden nun mit den von der jeweiligen Funktion oder Methode gelieferten Ergebnisdaten verglichen. Stimmt das erwartete Ergebnis mit dem gelieferten Ergebnis der Funktion oder Methode überein, so gilt der Test als bestanden.

Da bei dem Prestan Projekt keinerlei Unit-Tests eingesetzt wurden und das ein nachträglicher umfangreicher Test den Zeitrahmen dieser Diplomarbeit sprengen würde, beschränken sich die Unit-Tests nur auf die im Rahmen dieser Arbeit implementierten Erweiterungen. Für jede Erweiterung wurde eigens ein Testdatei geschrieben.

1. **Unit-Test XML-Ausgabe**
   Die Testdatei testmainxml.c ruft die einzelnen Funktionen der xml.c auf und versorgt diese mit den entsprechenden Parametern.

```
int main(int argc, char *argv[])
{
    open_xml_file();

    write_row_cell_s_xml_file("teststring1", 2);
    write_row_cell_s_xml_file("teststring2", 3);
    write_row_cell_s_xml_file("teststring3", 4);

    write_row_2cell_sn_xml_file("teststring4", 4, 6);
    write_row_2cell_sn_xml_file("teststring5", 5, 7);
    write_row_2cell_sn_xml_file("teststring6", 6, 8);

    write_row_2cell_ss_xml_file("teststring7", "value7", 10);
    write_row_2cell_ss_xml_file("teststring8", "value8", 11);
    write_row_2cell_ss_xml_file("teststring9", "value9", 12);

    close_xml_file();

    return 0;
}
.
```

Listing 4.15: Unit-Test: XML-Ausgabe

Das hier nachfolgende Listing zeigt das korrekte Ergebnis des Unit-Tests der XML-Ausgabe.

```
<?xml version="1.0"?>
<Workbook xmlns="urn:schemas-microsoft-com:office:spreadsheet"
xmlns:x="urn:schemas-microsoft-com:office:excel"
xmlns:ss="urn:schemas-microsoft-com:office:spreadsheet">
 <Worksheet ss:Name="Table1">
  <Table>
```

```
   <Row ss:Index="2">
    <Cell><Data ss:Type="String">teststring1</Data></Cell>
   </Row>
   <Row ss:Index="3">
    <Cell><Data ss:Type="String">teststring2</Data></Cell>
   </Row>
   <Row ss:Index="4">
    <Cell><Data ss:Type="String">teststring3</Data></Cell>
   </Row>
   <Row ss:Index="6">
    <Cell><Data ss:Type="String">teststring4</Data></Cell>
    <Cell><Data ss:Type="Number">4</Data></Cell>
   </Row>
   <Row ss:Index="7">
    <Cell><Data ss:Type="String">teststring5</Data></Cell>
    <Cell><Data ss:Type="Number">5</Data></Cell>
   </Row>
   <Row ss:Index="8">
    <Cell><Data ss:Type="String">teststring6</Data></Cell>
    <Cell><Data ss:Type="Number">6</Data></Cell>
   </Row>
   <Row ss:Index="10">
    <Cell><Data ss:Type="String">teststring7</Data></Cell>
    <Cell><Data ss:Type="String">value7</Data></Cell>
   </Row>
   <Row ss:Index="11">
    <Cell><Data ss:Type="String">teststring8</Data></Cell>
    <Cell><Data ss:Type="String">value8</Data></Cell>
   </Row>
   <Row ss:Index="12">
    <Cell><Data ss:Type="String">teststring9</Data></Cell>
    <Cell><Data ss:Type="String">value9</Data></Cell>
   </Row>

  </Table>
 </Worksheet>
</Workbook>.
```

Listing 4.16: Unit-Test: Ergebnisse der XML-Ausgabe

2. **Unit-Test ACP Erweiterung**
   Wie bei dem Unit-Test der XML-Ausgabe ruft die Testdatei testmainacl.c einzelne Funktionen der acl.c auf. Dabei zeigt das folgende Listing explizit einen Test für das Erzeugen eines ACL-Request Body.

```
#include <stdio.h>
#include <stdlib.h>
#include "testacl.h"

int main(int argc, char *argv[])
{

    printf("\nSet ACL Body %i\n", create_set_acl_body());

    printf("\nGet ACL Body %i\n", create_get_acl_body());

    return 0;
}

.
```

Listing 4.17: Unit-Test: ACP Erweiterung - ACL Bodys

Auch hier sei das korrekte Ergebnis des Unit-Tests für das Erzeugen eines ACL-Request Body aufgezeigt.

```
<?xml version="1.0" encoding="utf-8"?>
<acl xmlns='DAV:'>
 <ace>
  <principal>
   <href>http://www.example.com/users/esedlar</href>
  </principal>
  <grant>
```

```
    <privilege><read/></privilege>
    <privilege><write/></privilege>
   </grant>
  </ace>
  <ace>
   <principal>
    <property><owner/></property>
   </principal>
   <grant>
    <privilege><read-acl/></privilege>
    <privilege><write-acl/></privilege>
   </grant>
  </ace>
  <ace>
   <principal><all/></principal>
   <grant>
    <privilege><read/></privilege>
   </grant>
  </ace>
</acl>


<?xml version="1.0" encoding="utf-8"?>
<propfind xmlns='DAV:'>
 <prop>
  <acl xmlns='DAV:'/>
 </prop>
</propfind>

.
```

Listing 4.18: Unit-Test: Ergebnisse - ACP erweiterung - ACL Body

4.5.2 Black-Box-Test

Wie schon in der Systemanalyse, wird auch hier die Black-Box dazu verwendet, um zu zeigen, wie sich die Daten-Ausgabe (Output) im Bezug auf die Daten-Eingabe (Input) verhält. Damit kann kontrolliert werden, ob die ausgegeben Ergebnisse den der erwarteten Ergebnissen entspricht.

Abbildung 4.8: Black-Box-Test - Softwaretest

Die folgenden Tabellen zeigen die korrekten Black-Box-Test Ergebnisse für die Prestan Test Suite.

Input nur auf der Kommandozeile	
Hilfeaufruf	Prestan –help
Programmaufruf	Prestan bzw. ./Prestan
Parameter und Optionen für die URI	Protokoll (http) Zieladresse Port Pfad user password -r Anzahl der Requests -p Anzahl der Dead Properties -d Tiefe der Struktur von Collection -w Anzahl der Collection nach dem erreichen der Tiefe -o Name der Ausgabedatei -m Typ der WabDAV Methode

Tabelle 4.13: Black-Box-Test - Softwaretest - Ergebnisse Input

Output auf der Kommandozeile, in einer Logdatei oder XML-Datei	
Prestan Hilfe	Usage: ./Prestan [http://]hostname[:port]/path + [username password] [options] Option: -r, - -Requests Number of repeat requests (Default: 10) -p, - -Properties Number of dead properties (Default: 10) -d, - -Depth Depth of collection (Default: 10) -w, - -Width Width of collection at the bottom level(Default: 100) -o, - -Output Output file -m, - -Methods Type of Web Methods (WebDAV/WebFolder, Default:WebDAV) Example: ./Prestan http://dav.cse.ucsc.edu:81/basic test1 test1 -r 20 -p 20 -m WebFolder

Tabelle 4.14: Black-Box-Test - Softwaretest - Ergebnisse Output1

Output auf der Kommandozeile, in einer Logdatei oder XML-Datei	
Methoden und Ergebnisse	ProppatchMult ProppatchSingle PropfindDeadMult PropfindDeadSingle PropfindLiveMult PropfindLiveSingle Put1K Get1K Put64K Get64K Put1024K //new Get1024K //new Put2048K //new Get2048K //new Put3072K //new Get3072K //new Put4096K //new Get4096K //new Put5120K //new Get5120K //new Copy Move Delete MkCol CopyCol MoveCol DeleteCol Lock] UnLock LockCol UnLockCol GetAclSingleRes //new SetAclSingleRes //new GetAclSingleColl //new SetAclSingleColl //new GetAclMultiColl //new SetAclMultiColl //new

Tabelle 4.15: Black-Box-Test - Softwaretest - Ergebnisse Output2

4.5.3 Netzwerk Protokoll Analyse

Ein wichtiges Werkzeug bei dem Test von Client-Server Anwendungen ist die Netzwerk-Protokoll-Analyse. Mit ihrer Hilfe kann der Datenverkehr in einem Netzwerk untersucht werden. Dazu werden so genannte "Sniffer" bzw. "Network Packet Analyzer" eingesetzt.

Um nun zu zeigen, dass der Datenaustausch zwischen Client (Prestan) und Server korrekt ist und demzufolge die Implementierungen der Erweiterungen richtig verlaufen sind, wird hier ein solcher Network Packet Analyzer eingesetzt. *Wireshark* stellt einen solchen Analyzer dar.

Wireshark

Wireshark (ehemals Ethereal) ist ein Programm mit dem der Datenverkehr eines Netzwerks empfangen, aufgezeichnet, dargestellt und ausgewertet werden kann. Da Wireshark ein Open-Source Projekt ist, konnte es sich innerhalb weniger Jahren, vom *Do-it-yourself*-Programm zum führenden Netzwerkanalyse-Tool entwickeln. Die große Stärke von Wireshark liegt in der Unterstützung von zahlreichen Protokoll-Decodern [26].

Nachfolgend werden die korrekten Ergebnisse von zwei getesteten Funktionen dargestellt. Die restlichen Ergebnisse befinden sich im Anhang.

1. Get ACL Methode (Request, Response)

```
PROPFIND /slide/files/davtest/acl-test/aclsingleres HTTP/1.1
Host: 192.168.27.135:8080
User-Agent: davtest/0.1.0 neon/0.24.0-dev
Connection: TE
TE: trailers
Depth: 0
Content-Length: 114
Authorization: Basic cm9vdDpyb290
X-Prestan: (null): 2 (acl)


<?xml version="1.0" encoding="utf-8"?>
<propfind xmlns='DAV:'>
 <prop>
  <acl xmlns='DAV:'/>
 </prop>
</propfind>

.
```

Listing 4.19: Wireshark: Get ACL Methode - Request

```
HTTP/1.1 207 Multi-Status
Pragma: No-cache
Cache-Control: no-cache
Expires: Thu, 01 Jan 1970 01:00:00 CET
Set-Cookie: JSESSIONID=B832B35263BF0A8DCC67959C89EF23B9; Path=/slide
Content-Type: text/xml;charset=UTF-8
Transfer-Encoding: chunked
Date: Thu, 05 Jul 2007 18:19:33 GMT
Server: Apache-Coyote/1.1


<?xml version="1.0" encoding="UTF-8"?>
<D:multistatus xmlns:D="DAV:">
<D:response>
 <D:href>/slide/files/davtest/acl-test/aclsingleres</D:href>
  <D:propstat>
   <D:prop>
```

```
<D:acl>
  <D:ace>
   <D:principal>
    <D:unauthenticated />
   </D:principal>
   <D:grant>
    <D:privilege>
     <D:all />
    </D:privilege>
   </D:grant>
   <D:inherited>
    <D:href>/slide/files</D:href>
   </D:inherited>
  </D:ace>
  <D:ace>
   <D:principal>
    <D:href>/slide/roles/user</D:href>
   </D:principal>
   <D:grant>
    <D:privilege>
     <D:write />
    </D:privilege>
   </D:grant>
   <D:inherited>
    <D:href>/slide/files</D:href>
   </D:inherited>
  </D:ace>
  <D:ace>
   <D:principal>
    <D:property>
     <D:owner />
    </D:property>
   </D:principal>
   <D:grant>
    <D:privilege>
     <D:read-acl />
    </D:privilege>
   </D:grant>
   <D:inherited>
    <D:href>/slide/files</D:href>
   </D:inherited>
  </D:ace>
  <D:ace>
   <D:principal>
    <D:href>/slide/roles/root</D:href>
   </D:principal>
   <D:grant>
    <D:privilege>
     <D:all />
    </D:privilege>
   </D:grant>
   <D:inherited>
    <D:href>/slide/</D:href>
   </D:inherited>
  </D:ace>
  <D:ace>
   <D:principal>
    <D:all />
   </D:principal>
   <D:deny>
    <D:privilege>
     <D:read-acl />
    </D:privilege>
    <D:privilege>
     <D:write-acl />
    </D:privilege>
    <D:privilege>
     <D:unlock />
    </D:privilege>
   </D:deny>
   <D:inherited>
    <D:href>/slide/</D:href>
   </D:inherited>
  </D:ace>
  <D:ace>
   <D:principal>
    <D:all />
   </D:principal>
   <D:grant>
    <D:privilege>
     <D:read />
    </D:privilege>
   </D:grant>
   <D:inherited>
    <D:href>/slide/</D:href>
   </D:inherited>
  </D:ace>
```

```
    </D:acl>
   </D:prop>
   <D:status>HTTP/1.1 200 OK</D:status>
  </D:propstat>
</D:response>
</D:multistatus>

.
```

Listing 4.20: Wireshark: Get ACL Methode - Response

2. Set ACL Methode (Request, Response)

```
ACL /slide/files/davtest/acl-test/aclsingleres HTTP/1.1
Host: 192.168.79.135:8080
User-Agent: davtest/0.1.0 neon/0.24.0-dev
Connection: TE
TE: trailers
Depth: 0
Content-Length: 529
Authorization: Basic cm9vdDpyb290
X-Prestan: (null): 2 (acl)


<?xml version="1.0" encoding="utf-8"?>
<acl xmlns='DAV:'>
 <ace>
  <principal>
   <href>http://www.example.com/users/esedlar</href>
  </principal>
  <grant>
   <privilege><read/></privilege>
   <privilege><write/></privilege>
  </grant>
 </ace>
 <ace>
  <principal>
   <property><owner/></property>
  </principal>
  <grant>
   <privilege><read-acl/></privilege>
   <privilege><write-acl/></privilege>
  </grant>
 </ace>
 <ace>
  <principal><all/></principal>
  <grant>
   <privilege><read/></privilege>
  </grant>
 </ace>
</acl>

.
```

Listing 4.21: Wireshark: Set ACL Methode - Request

```
HTTP/1.1 200 OK
Pragma: No-cache
Cache-Control: no-cache
Expires: Thu, 01 Jan 1970 01:00:00 CET
Set-Cookie: JSESSIONID=BB7C6F6E022E54C168178B9CAA34E443; Path=/slide
Content-Length: 0
Date: Thu, 05 Jul 2007 18:19:34 GMT
Server: Apache-Coyote/1.1

.
```

Listing 4.22: Wireshark: Set ACL Methode - Response

Kapitel 5

Auswahl der WebDAV Server

5.1 Kriterien

Nach dem Abschluß der Erweiterung der Testsuite um eine ACL-Unterstützung stellt das zweite primäre Ziel dieser Arbeit die Findung einer Antwort auf die Frage dar, wie es um die Leistungsfähigkeit verschiedener WebDAV Implementierungen bestellt ist. Im Rahmen von Leistungstests sollen dabei vor allem auch ACL- betreffende Aspekte in die Betrachtungen mit einbezogen werden. Kapitel 2 hat bereits den großen Umfang an verfügbaren WebDAV Implementierungen aufgezeigt. Da eine Betrachtung einer jeden Implementierung den Rahmen dieser Arbeit sprengen würde, soll im Vorfeld eine wohl definierte Selektion zu testender Implementierungen erfolgen, welche durch die Festlegung von Kritierien umgesetzt wird.

Um einen konkreten Ansatz zu finden, nach welchen Kriterien die Auswahl erfolgen soll, muss zunächst einmal geklärt werden, welche Bedingungen zu unterschiedlichen Leistungen eines Servers führen können. Dabei soll angenommen werden, dass die verwendete Hardware, auf welchen die Server ausgeführt werden, immer gleich ist.

Nach einer hinreichenden Analyse ergeben sich folgende Kriterien, die zu einer unterschiedlichen Leistung eines Servers führen können.

- Betriebssystem (Windows, Linux)
- Datenverwaltung (Filesystem, Datenbank)
- WebDAV Erweiterungen (DeltaV, DASL, ACP)
- Art der Implementierung (Programmiersprache und Weiteres)

Neben den genannten Kriterien, welche unmittelbar die Leistungsfähigkeit betreffen, sollen auch anderweitige Kriterien zur Durchführung einer Selektion herangezogen werden, welche der folgenden Auflistung zu entnehmen sind.

- Kostenfrage (Lizenzen und Weiteres)
- Verbreitungsgrad der Server
- schon in Verwendung am DLR Standort

5.2 Auswahl

Eine Anwendung der Kritierenliste auf die Menge der verfügbaren WebDAV Server legt die nachfolgend aufgelisteten Implementierungen für eine Einbeziehung in die durchzuführenden Leistungstests fest. Bevor die Leistungstests durchgeführt werden, sollen zunächst die Charakteristika der einzelnen WebDAV Server herausgearbeitet werden. Zudem soll den Tests eine kurze Beschreibung einer jeden Implementierung vorausgehen.

Apache2 mit mod_dav

- ist einer der meist genutzten Server (>60% NetCraft [27])
- kostenfrei, Open-Source, Apache Lizenz
- läuft sowohl unter Windows, als auch unter Linux
- unterstützt DeltaV mit dem Modul mod_dav_svn

Catacomb

- ist eine Erweiterung von Apache2
- Datenbank (MySQL)
- läuft ausschließlich unter Linux
- unterstützt DeltaV, DASL und ACP
- wird am Standort des DLR weiterentwickelt

Jakarta Slide

- gehört zur Apache Software Foundation
- basiert auf Java -> Plattformunabhängig
- verfügt über eine unabhängige Datenverwaltung (wahlweise Filesystem, Datenbank...)

Lighttpd

- steht in direkter Konkurrenz zum Apache2 Server bei den Open-Source Projekten
- sehr gute Performance

- läuft ausschließlich unter Linux

Microsoft Internet Information Server 5.1 (IIS5.1)

- IIS Familie steht an zweiter Stelle von den meist genutzten Server (>15% NetCraft [27])
- unterstützt Active Server Pages (ASP) und mit ISAPI-Filtern auch PHP und JSP
- läuft ausschließlich unter Windows

Microsoft SharePoint

- wird schon am Standort der DLR eingesetzt
- weist eine hohe Komplexität auf und hat leistungsstarke Funktionen
- läuft ausschließlich unter Windows 2003 Server

Tamino WebDAV Server

- besitzt dieselben Stärken wie der Catercomb (Unterstützung für DeltaV, DASL und ACP)
- verwaltet die Daten in einer XML-Datenbank

5.2.1 Apache2 und mod_dav

Der Apache Webserver ist eine Weiterentwicklung des *NCSA Webservers*, welcher ursprünglich am *National Center for Supercomputing Applications (NCSA)* der Universität von Illinois von Rob McCool entwickeltet wurde.

Nachdem Rob McCool das NCSA 1994 verlassen hatte, wurde die Entwicklung dieses Webservers zunächst nicht weitergeführt. Verschiedene Institutionen und Einzelpersonen haben jedoch zahlreiche Patches entwickelt. Diese Patches wurden von den Begründern des Apache Projektes gesammelt und mit dem ursprünglichen Quellcode zu einer einheitlichen Distribution zusammengefasst. So entstand im Februar 1995 mit der Gründung der *Apache Group* der Webserver *Apache* (“a patchy web server”) [Gie04].

Einer der Gründe für den Siegeszug war die Entwicklung der Modultechnologie, die es ermöglichte, auf relativ einfache Weise zusätzliche Funktionalitäten in den Webserver zu integrieren. So ließ sich der Webserver beispielsweise um eine PHP- und Perl-Unterstützung erweitern. Ein weiter Grund für die rasche Verbreitung war auch die Open-Source-Lizenz, unter welcher der Apache Webserver zur Verfügung gestellt wurde.

Der Apache Webserver ist bereits ein Jahr nach seiner Veröffentlichung zu einer der meistgenutzten Webserver geworden. Zunächst war lediglich ein Betrieb

auf Unix-basierenden Systemen gegeben, was sich jedoch mit der Entwicklung der zweiten Version änderte. So konnte die im April 2002 freigegebene Version 2.0.35 erstmals auch als stabile Version für Windows-Systeme bezeichnet werden.

Im März 1999 entschlossen sich die Mitglieder der Apache Group die *Apache Software Foundation (ASF)* zu gründen; eine gemeinnützige Organisation mit Sitz in den USA. Zum Apache Projekt sind im Laufe der Zeit mehr als 20 weitere Open-Source Projekte hinzugekommen [28].

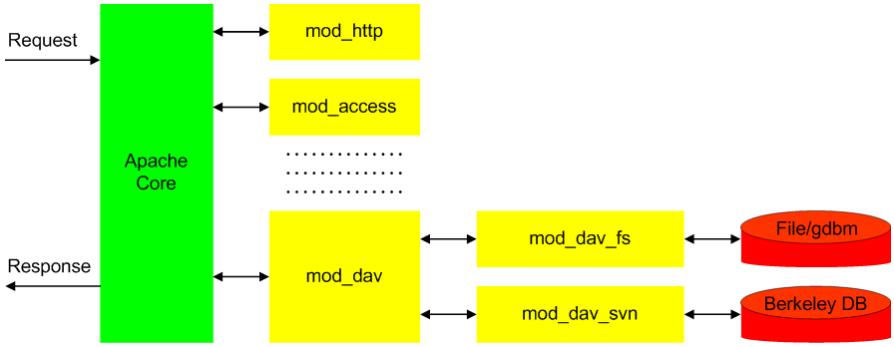

Abbildung 5.1: Apache2: Architektur mit mod_dav und mod_dav_svn

5.2.2 Catocomb

Catacomb ist ein WebDAV Modul für den Apache Webserver und wurde an der Universität von Kalifornien unter der Leitung von Prof. Jim Whitehead entwickelt. Catacomb erweitert das Standardmodul mod_dav um einige Zusatzfunktionen des WebDAV Protokolls.

Das Apache Standardmodul mod_dav wird normalerweise durch das Modul mod_dav_fs erweitert, das eine Schnittstelle zwischen mod_dav und dem Dateisystem schafft. Sowohl Inhalt als auch die Metadaten der Ressource werden dabei im lokalen Dateisystem gespeichert.

Catacomb hingegen ersetzt das Modul mod_dav_fs durch mod_dav_repos und speichert alle Metadaten bzw. die Ressourcen (wählbar) in einer relationalen Datenbank (MySQL) . Bekanntlich ist das Suchverfahren von vielen Ressourcen und deren Metadaten bis zu einer bestimmten Größe in einer Datenbank wesentlich effizienter, als das eines lokalen Dateisystems. Zudem vereinfacht die Verwendung einer Datenbank auch die Umsetzung der WebDAV Erweiterungen für die serverseitige Suche (DASL) und die der Versionisierung von Ressourcen (DeltaV) [29].

Das Catacomb Projekt hat als erstes Open-Source Projekt die WebDAV Erweiterungen DASL und DeltaV implementiert. Des Weiteren wurde Catacomb von Markus Litz [Lit06] (beim DLR) um das Access Control Protocol (ACP) erweitert.

Nachfolgend werden noch einmal alle Erweiterungen des WebDAV Protokolls, welche von Catacomb unterstützt werden, aufgeführt:

- Klasse 1, 2 Fähigkeiten des RFC 2518
- eine beliebige Anzahl von benutzerdefinierten dead properties
- volle Unterstützung der live properties aus RFC 2518
- DASL Protokoll
- Versionisierung aus dem RFC 3253 mit Unterstützung der HTTP-Methoden: VERSION-CONTROL, CHECKIN, CHECKOUT, UNCHECKOUT, REPORT.
- die Möglichkeit, den Inhalt einer Ressource und deren Metadaten getrennt zu speichern
- Access Control Protocol (ACP) nach RFC 3744

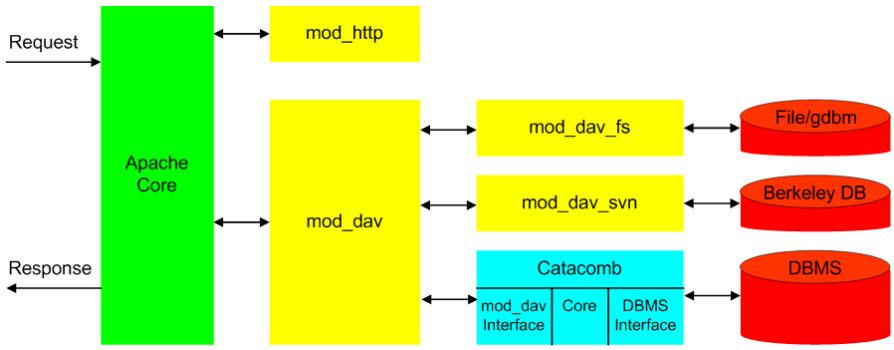

Abbildung 5.2: Catacomb: Architektur

5.2.3 Jakarta Slide

Slide ist ein auf Java basierendes Framework zur Entwicklung von Content Management Systemen und Bestandteil des Open-Source Projektes Jakarta, welches von der Apache Software Foundation (ASF) und Sun gegründet wurde.

Die Architektur von Slide besteht aus einer *Matrix von Modulen*, wobei das Hauptmodul ein Content Management und Integrationssystem darstellt, das

als Content Management Framework angesehen werden kann [PR02].

Jakarta Slide bietet eine Client-API an, die für eigene Projekte genutzt werden kann und es unterstützt die hierarchische Organisation von Daten und deren Speicherung in beliebige, verteilte Verzeichnisse. Dies wird durch die Abstraktion der Daten ermöglicht. Das heißt, dass die Daten extern gespeichert werden, während Slide ausschließlich mit Abbildungen der Daten arbeitet (abstraction layer). Neben einer Abspeicherung im lokalen Dateisystem können Datenbanken verschiedener Hersteller oder freier Projekte als Ablagesystem genutzt werden.

Ein weiteres Zugangsmodul unterstützt das WebDAV Protokoll und seine Erweiterungen wie DeltaV, DASL (DAV Searching and Locating) und ACP. Es macht Slide zu einer idealen Wahl für netz-basierendes Content Management. Auf alle von Slide verwalteten Inhalte kann demnach auch über WebDAV zugegriffen werden.

Slide ist äußerst vielseitig verwendbar und lässt sich vor allem im Intranet zur Applikations- oder Fileserververwaltung einsetzen [19].

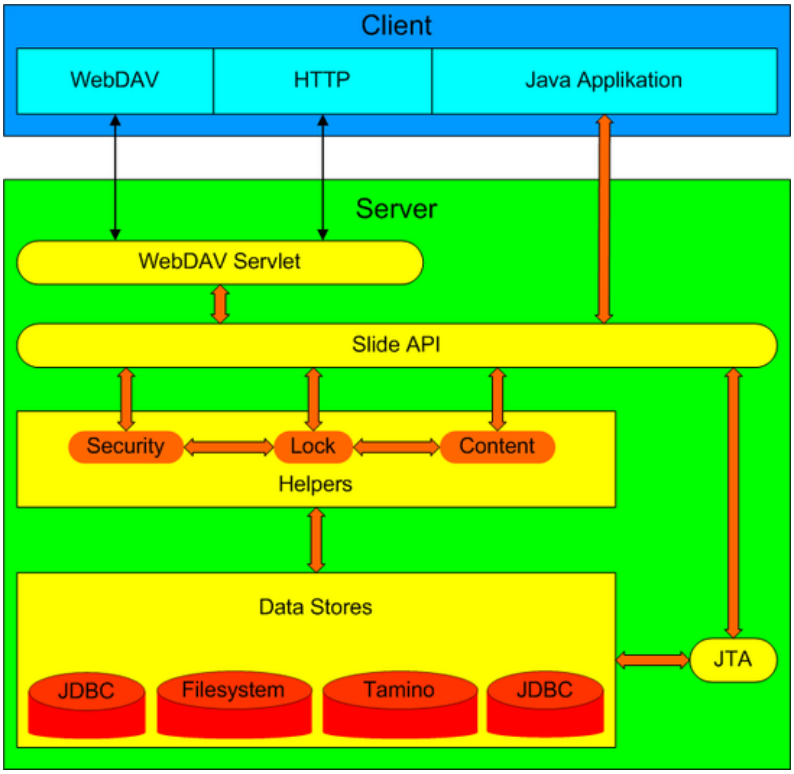

Abbildung 5.3: Jakarta Slide: Architektur

5.2.4 Lighttpd

Das Projekt des *Ligthhtpd* Servers wurde 2003 von Jan Kneschke initiiert. Das Konzept sah vor, einen schlanken und schnellen Webserver zu entwerfen.

Durch immer wieder eingebrachte Optimierungen hat sich Lighttpd zu einem sicheren, schnellen und sehr flexiblen Webserver für Hochleistungsumgebungen entwickelt. Dabei werden die wichtigsten Funktionen des Webservers - ähnlich wie beim Apache Servers - durch die Implementierung von Modulen erweitert.

Durch seinen niedrigen Speicherplatzbedarf und dem effektiven Management des benötigten Bedarfs an Rechenzeit ist ein hoher Datendurchsatz - auch bei vielen (>1000) parallelen Verbindungen - gewährleistet.

Durch die Verwendung eines integrierten *Load-Balancers* lassen sich mehrere externe FastCGI-Prozesse (wie z.B. PHP, Perl, ...) unter optimaler Auslastung an den Webserver koppeln. Damit eignet sich der Lighttpd besonders für Server, die viele parallele Anfragen zu beantworten haben und deshalb einer stetig hohen CPU-Belastung unterliegen [21].

- geringer Speicherverbrauch (auch bei 1000 parallelen Verbindungen im Schnitt nur 32 MB)
- hoher Durchsatz (um den Faktor 4-6 schneller als ein Apache-Webserver)
- hohe Stabilität auch bei vielen parallelen Anfragen

Die hervorragende Performance des Lighttpd Servers ist mittlererweile in einer Vielzahl von Leistungstests dokumentiert. Ein Teil dieser Benchmarks kann dabei unter `http://www.lighttpd.net/benchmark` [30] eingesehen werden.

Der Lighttpd Server ist ein Open-Source Projekt (BSD-Lizenz) und erfreut sich mittlerweile immer größerer Beliebtheit (vgl. NetCraft [27])

5.2.5 Microsoft Internet Information Server

Der Microsoft Internet Information Server (IIS) ist ein Zusammenschluss von verschiedenen Internet Informationsdiensten, welche für die Microsoft Betriebssysteme entwickelt wurden. Er zeichnet sich laut Microsoft durch hohe Leistungsfähigkeit, Integration in Windows-Server und eine einfache Administrierbarkeit aus und steht laut NetCraft [27] weltweit an zweiter Stelle der beliebtesten Webserver. Die wichtigsten Informationsdienste sind unter anderem die Webserver-, FTP-, SMTP-, POP3-, WebDAV- und Index-Dienste.

Der IIS kann Active Server Pages (ASP) oder .NET-Applikationen (ASP.NET) ausführen. Durch die Installation von sogenannten ISAPI-Filtern lassen sich auch PHP- und JSP-basierende Webseiten betreiben.

Standardmäßig wird der IIS mit den Serverversionen von Windows NT 4.0 (IIS 4), Windows 2000 (IIS 5), Windows XP Professional (IIS 5.1) und Windows Server 2003 (IIS 6) ausgeliefert.

Die aktuelle Version (IIS 7) wird im Rahmen von Windows Vista und Windows Server 2008 erscheinen [22].

5.2.6 Microsoft SharePoint Portal Server

Das SharePoint Konzept sieht vor, dass die IT-Welt im Unternehmen stärker zusammenwächst und dass für die elektronische Zusammenarbeit die Integration der Windows SharePoint Services und die der SharePoint Portal Server ein wesentlicher Bestandteil wird.

Mithilfe der kombinierten Funktionen des SharePoint Services und der SharePoint Portal Server kann ein Unternehmen ein intelligentes Portal entwickeln, das einzelne Mitarbeiter, Teams und die Erfahrungen aus unterschiedlichen Geschäftsprozessen miteinander verbindet.

Die SharePoint-Grundstruktur ist auf den ersten Blick eine Webanwendung. Näher betrachtet, beinhaltet das SharePoint Konzept viele komplexe und leistungsstarke Funktionen, die sich grob in drei Komponenten einteilen lassen [31].

- **Webinterface:** SharePoint ist aus Sicht des Benutzers eine Webanwendung. Die primäre Nutzung erfolgt dabei über einen Browser, was jedoch keine Grundvoraussetzung darstellen muß.
- **SharePoint-Dienste:** Sind alle nicht unmittelbar ersichtlichen Funktionen der Softwarekomponenten (Zugriff auf Listen und Bibliotheken, Ermittlung von Benutzergruppen, Benachrichtigung von Benutzern, der Suchdienst und andere).
- **Datensicherung:** Gespeichert werden die Konfigurationsdaten und auch die eigentlichen Nutzdaten der Benutzer in einer Datenbank. Die Datenbank muss ein Microsoft SQL Server sein.

Für die gesamte SharePoint-Kommunikation zwischen den Servern und Clients werden folgende Protokolle verwendet:

- HTTP und WebDAV
- HTTPs (SSL/TLS)
- SMB (Server Message Block)

Wichtig!

Der SharePoint Service und der SharePoint Portal Server laufen auf dem Windows 2003 Server Betriebssystem und benutzen den Internet Information Server 6.0 (ISS6.0) als Webserver. Dabei werden alle WebDAV Funktionalitäten des ISS6.0 übernommen und mit den Funktionalitäten von SharePoint erweitert.

5.2.7 Tamino WebDAV Server

Der *Tamino WebDAV Server* ist eine kommerzielles Produkt der Software AG. Es ist eine Hauptkomponente der Tamino XML Server Architektur und stellt den einfachen Zugriff auf die Daten sicher. Der Server ermöglicht die gemeinsame Bearbeitung von Inhalten durch mehrere Autoren über das Intranet und Internet.

Der Tamino WebDAV Server dient dabei als Schnittstelle zwischen Client-Anwendungen (Office-Produkten oder Browsern) und dem Tamino XML Server als Backend Data Store.

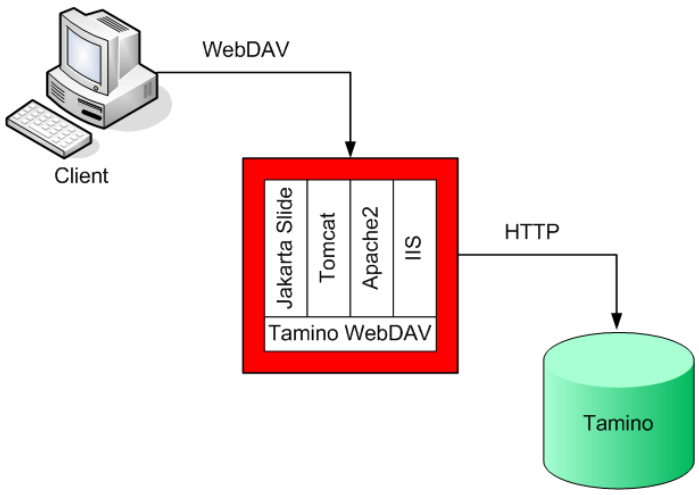

Abbildung 5.4: Tamino: WebDAV Architektur

Die jetzige Version (4.4.1) des Tamino WebDAV Servers besitzt alle aktuellen WebDAV Erweiterungen (bsw. DeltaV, DASL und ACP) und erweitert somit die Funktionalität von Tamino XML Server um Namespace-Management (verschachtelte Erfassungen oder Verzeichnisse), zusätzliche Eigenschaften (z.B. Datum der letzten Änderung, Länge des Inhalts oder Typ des Inhalts) und Überschreibschutz (dauerhafte Sperrung) [23].

Wichtig!

Wie Abbildung 5.4 zeigt, nutzt Tamino WebDAV, wie auch schon der Microsoft SharePoint Portal Server, keinen eigenen Webserver. Tamino stellt lediglich ein eigenes WebDAV Modul bereit, dass in den jeweiligen Webserver integriert wird.

Je nach Betriebssystem, könne folgende Webserver verwendet werden:

- Jakarta Slide,
- Tomcat,
- Apache,
- oder IIS.

Für den Leistungstest wird das ein Apache2-Webserver mit Tamino Starter-Kit verwendet, wobei Windows XP Professional als Betriebssystem verwendet werden wird. Die Integration des Modules erfolgt über eine Modifikation der Konfigurationsdatei des Apache-Webservers. Das nachfolgende Listing bildet exemplarisch den wesentlichen Ausschnitt einer solchen Konfigurationsdatei ab.

```
# Software AG # Start # System Management Hub #
# 3.6.1.10 - 2005-09-27-14.09.20
ScriptAliasMatch  ^/smh/images2/(.*)
"C:/Programme/Software AG/System Management Hub/bin/argmlcgi.exe/$1"

ScriptAliasMatch  ^/smh/help/(.*)
"C:/Programme/Software AG/System Management Hub/bin/argmlcgi.exe/$1"

ScriptAlias       /smh/cgi/
"C:/Programme/Software AG/System Management Hub/bin/"

Alias             /smh/
"C:/Programme/Software AG/System Management Hub/htmllayer/"
# Software AG #  End  # System Management Hub #


# Start Tamino version
# ------------------------------------------------------------------
# Software-AG--Tamino----
LoadModule  ino_module modules/Apache2ModuleIno.dll

<Location /tamino>
SetHandler ino
</Location>

<Location /tamino_no_challenge>
SetHandler ino
NoChallenge true
</Location>
```

```
<LocationMatch /tamino\.>
SetHandler ino
</LocationMatch>
# ------------------------------------------------------------------
# End Tamino version


# Start Tamino Alias
# ------------------------------------------------------------------
# Software-AG--Tamino----
Alias   /tamino-v4.4.1.1/en/Documentation
"C:/Programme/Software AG/Tamino/Tamino
4.4.1.1/Documentation/en"

Alias   /tamino-v4.4.1.1/en/welcome
"C:/Programme/Software AG/Tamino/Tamino
4.4.1.1/WelcomeToTamino/en"

Alias   /tamino-v4.4.1.1/en/tii
"C:/Programme/Software AG/Tamino/Tamino
4.4.1.1/X_Tools/Tamino_Interactive_Interface/en"

Alias   "/System Management Hub/help/doc"
"C:/Programme/Software AG/System Management Hub/help/doc"

Alias   /tamino-v4.4.1.1/README.TXT
"C:/Programme/Software AG/Tamino/Tamino 4.4.1.1/readme.txt"
# ------------------------------------------------------------------
# End Tamino Alias


# ------------------------------------------------------------------
# Enable Apache Web Server Authentication for Tamino Security
# Note: The AuthUserFile is created with pgm ../apache/bin/htpasswd.exe
#       Enter command 'htpasswd -h' for pgm usage description
#       Restart Apache Web Server after modification
# If authorization fails:
# - Check Apache error.log file
# - Enter browser command http://localhost/tamino/secdb?_diagnose=echo

<Location /tamino/welcome_4_4_1>
AllowOverride AuthConfig
AuthName "Tamino"
AuthType Basic
AuthUserFile "userids"
require valid-user
</Location>

.
```

Listing 5.1: Tamino WebDAV: Apache2 Konfigurationsdatei

5.3 Zusammenfassung

Nachdem die Server vorgestellt wurden, werden in der nachfolgenden Tabelle alle wesentlichen Spezifikationen der Server abchließend noch einmal zusammengefasst.

Server/ Funktion	Apache mit mod_dav	Catacomb	Jakarta Slide	Lighttpd	Microsoft IIS5.1	Microsoft SharePoint	Tamino
WebDAV	√	√	√	√	√	√	√
Klasse	1, 2	1, 2	1, 2	1, 2	1, 2	1,2	1, 2
DeltaV	√	√	√	-	√	√	√
DASL	-	√	√	-	-	√	√
ACP	-	√	√	-	-	√	√
Datenverwaltung	Filesystem	Datenbank (MySQL)	unabhängig	Filesystem	Filesystem	Datenbank (SQL, MSDE)	Datenbank
Kostenfrei	ja Open Source	ja Open Source	ja Open Source	ja Open Source	ja (begrenzt)	nein	nein
Lizenz	Apache-Lizenz	Apache-Lizenz	Apache-Lizenz	BSD-Lizenz			
Verbreitungsgrad	60%	?	?	?	über 15%	?	?
Betriebssystem	Windows Linux other	Linux	Linux	Linux	Windows	Windows	Windows Linux other
Beschreibung	DeltaV mit mod_svn						

Tabelle 5.1: Auswahl der WebDAV Server

Kapitel 6

WebDAV Server Leistungstests und Auswertung

6.1 Design und Entwickeln der Testumgebung

Nach der erfolgten Selektion und Beschreibung zu testender WebDAV Server sollen in diesem Kapitel die angestrebten Leistungstests durchgeführt und die Ergebnisse der Testreihen zur Auswertung gebracht werden. Betrachtet werden sollen dabei zwei essentielle Leistungsmerkmale; die Konformität der Implementierung in Bezug auf die definierten WebDAV-Standards und die Performanz hinsichtlich der Geschwindigkeit der Ausgaben. Die Messreihen werden von der Clientseite aus vorgenommen.

Um korrekte Ergebnisse erzielen zu können, müssen die Testreihen in einer homogenen Umgebung erfolgen, so dass die Bedingungen über alle Testreihen hinweg konstant bleiben, so dass vergleichbare Ergebnisse resultieren. Der Weg, welcher zu gehen ist, um diese homogene Umgebung zu schaffen, wird in den folgenden fünf Schritten dargestellt.

Schritt 1: Aufbau der Testumgebung

Der erste Schritt befasst sich mit dem Aufbau der Testumgebung. Ziel ist es, die Interaktion zwischen Client und Server möglichst ungestört verlaufen zu lassen, so dass der Latenzzeitenunterschied vernachlässigt werden kann.

Wie in der nachfolgenden Abbildung vereinfacht dargestellt wird, kann dies durch eine einfache drahtgebundene Verbindung zwischen zwei Computern erreicht werden.

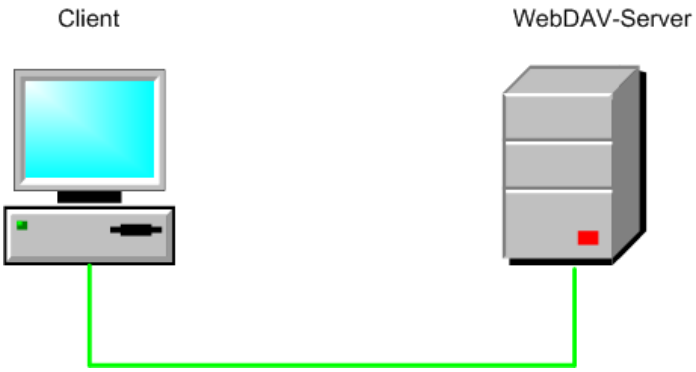

Abbildung 6.1: Aufbau der Test-Umgebung

Schritt 2: Hardware
Der zweite Schritt schließt die Bestimmung der für die Testumgebung verwendenden Hardware ein. Für alle Server werden die gleichen Systemkomponenten verwendet.

Systemhersteller	FUJITSU SIEMENS
Systemtyp	X86-basierter PC
Prozessor	x86 Family 15 Model 2 Stepping 9 GenuineIntel ~3192 Mhz
BIOS-Version	FUJITSU SIEMENS // Phoenix Technologies Ltd. 5.00 R2.10.1567.01
Gesamter realer Speicher	2.048,00 MB
Netzwerk-Adapter	Intel(R) PRO/1000 CT Network Connection

Abbildung 6.2: System-Hardeware der Testumgebung

Schritt 3: Software
Der dritte Schritt nimmt Bezug auf die verwendete Software. Für die Server werden die Betriebssysteme Microsoft Windows XP Professional mit Service Pack 2 (nachfolgend kurz als Win32 bezeichnet), Microsoft Windows 2003 Server und die Linux-Distribution *Ubuntu* verwendet.

Für die Messungen werden die schon in Kapitel 3 vorgestellten WebDAV Server Test Suites Litmus und Prestan benutzt. Diese Test Suites sind aus-

schließlich unter Linux lauffähig und liegen in zwei verschiedenen Ausführungen vor.

Für die Konformitätstests wird die Litmus Test Suite sowohl ohne ACP-Erweiterung, als auch mit ACP-Erweiterung verwendet. Selbiges gilt für die durchzuführenden Performancetests, wobei für diese Tests die Prestan Test Suite zum Einsatz kommt.

Schritt 4: Testdurchführung und Datenerfassung
Nach der Bestimmung des Aufbaus der Testumgebung, sowie der zu verwendeten Hard- und Software ist der Ablauf eines Tests festzulegen, so dass gewährleistet werden kann, dass alle Tests gleichermaßen verlaufen.

Im Vorfeld der Leistungstests wird einmalig die Testumgebung vorbereitet. Dazu werden beide Computer (Client und Server) fest positioniert, über ein Ethernet-Kabel miteinander verbunden und gestartet. Mit Hilfe des ping-Befehls wird anschließend die Konnektivität zwischen ihnen überprüft.

Nach erfolgreicher Installation und abgeschlossener Konfiguration des jeweils zu testenden WebDAV Servers erfolgt der eigentliche Leistungstest.

In einem ersten Schritt werden die WebDAV Server mit den Test Suites ohne ACP Erweiterung getestet. Im Anschluss erfolgt eine äquivalente Messreihe, welche eine vorhandene ACP Erweiterung einschließt.

Die Messergebnisse werden zentral in einem Verzeichnis gesammelt und so für die Auswertung bereitgestellt.

Schritt 5: Auswertung
Vergleichbar mit der Durchführung eines Tests muss auch die Vorgehensweise bei der Auswertung der Testergebnisse hinreichend fixiert werden.

Bevor die anstehenden Tests durchgeführt und ausgewertet werden, wird anhand der Spezifikation eines jeden WebDAV Servers eine kurze Einschätzung hinsichtlich der zu erwartenden Testergebnisse erstellt und in einer Tabelle zusammengefasst.

Standardmäßig geben beide Test Suites die Messergebnisse in einer Kommandozeile aus. Als Erweiterung und zum Zwecke der besseren Auswertung können diese Messergebnisse jedoch zusätzlich von den Test Suites in eine Logdatei geschrieben werden.

Um die Auswertung durch eine grafische Aufbereitung der Messdaten unterstützen zu können, wird auf die XML-Ausgabe der Prestan Test Suite zurück-

gegriffen. Durch eine zwischengelagerte Umformatierung der auf XML-Basis vorliegenden Ergebnisse kann eine Abbildung der Ergebnisse auf Balkendiagramme erfolgen, welche sich im weiteren Verlauf optimal gegenüberstellen lassen.

Zur Vervollständigung der Auswertung sollen die in den Balkendiagrammen zusammengefassten Testergebnisse abschließend diskutiert werden.

6.2 Leistungstest ohne Access Control Protocol

In dem nachfolgenden Abschnitt werden die zu erwartenden und gemessenen Ergebnisse des Leistungstests ohne Access Control Protocol dargestellt.

6.2.1 Litmus Konformitätstest

Der Konformitätstest, welcher ACP ausschließt, ist in fünf Testbereiche (basic, copymove, props, locks, und http) eingeteilt und umfasst in seiner Gesamtheit 84 unterschiedliche Tests. Das Ziel der Konformitätstests stellt die Gewinnung der Erkenntnis dar, inwiefern unterschiedliche WebDAV-Implementierungen sich an die in den Standards vereinbarten Vorgaben halten.

Test-Erwartung

- **Apache2:** Da der Apache2 Server einer der ersten Open-Source Server mit WebDAV Unterstützung war und die Entwickler eng mit den WebDAV-Organisationen zusammenarbeiten, wird eine hohe Konformität erwartet [32].
- **Catacomb ohne ACP Erweiterung:** Bei Catacomb handelt es sich ebenfalls um ein Open-Source Projekt. Weiterhin stellt Catacomb eine Erweiterung des Apache WebDAV Servers dar. Aus diesem Grunde wird auch hier eine hohe Konformität zu erwartet.
- **Lighttpd:** Die WebDAV-Funktionalität des Lighttpd Servers wird, vergleichbar mit Apache, durch die Erweiterung mit entsprechenden Modulen bereitgestellt. Da die Module jedoch mit Hilfe von erweiterten Bibliotheken an die jeweilige Plattform angepasst werden müssen, wird eine unvollständige Konformität erwartet.
- **Jakarta Slide:** Jakarta Slide ist - wie der Apache2 Server - ein Projekt, das zu der Apache Software Foundation gehört. Es kann davon ausgegangen werden, dass bei der Implementierung von WebDAV unter Java die gleichen Methoden verwendet wurden. So wird auch hier eine hohe Konformität erwartet.
- **Microsoft IIS5.1:** Der Microsoft Internet Information Server 5.1 unter Windows XP Pro. besitzt als kostenfreie, kommerzielle Anwendung nur eine minimale Konfigurierbarkeit, was auf eine unvollständige Konformität schließen lässt.
- **Microsoft SharePoint:** SharePoint ist ein kommerzielles Produkt von Microsoft und nutzt als Webserver den Microsoft IIS6.0. Dieser wird durch das SharePoint Konzept mit vielen komplexen und leistungsstarken Funktionen erweitert. Es wird eine hohe Konformität erwartet.
- **Tamino ACP deaktiviert:** Der Tamino WebDAV Server ein weiteres kommerzielles Produkt. Vergleichbar mit SharePoint, besitzt auch Tamino keinen eigenen Webserver. Er fügt in der jetzigen Version einem

bestehenden Webserver ein eigenes Modul mit allen aktuellen WebDAV Erweiterungen hinzu. Die ACP Erweiterung kann aktiviert oder deaktiviert werden. Es wird eine hohe Konformität erwartet, da die WebDAV-Unterstützung das Alleinstellungsmerkmal von Tamino darstellt.

Test-Ergebnisse

In den Tabellen 6.1 und 6.2 werden alle gemessenen Ergebnisse des Konformitätstests ohne Access Control Protocol dargestellt.

Server -> Methode Name/Anzahl		Apache2 Linux	Apache2 Win32	Catacomb Linux	Lighttpd Linux	Jakarta Slide2.1 Linux
basic	14	14	14	14	14	14
copymove	11	11	11	11	11	11
props	25	25	25	25	25	25
locks	30	30	30	30	27	29
http	4	4	4	4	4	4
Summe:	84	84	84	84	81	83
Prozent:		100,0%	100,0%	100,0%	96,4%	98,8%

Tabelle 6.1: Litmus ohne ACP: Ergebnisse 1

Server -> Methode Name/Anzahl		Jakarta Win32	Microsoft IIS5.1 Win32	Microsoft SharePoint 2003 Server	Tamino Win32
basic	14	14	14	14	14
copymove	11	11	11	11	11
props	25	25	20	20	24
locks	30	29	-	26	24
http	4	4	4	4	4
Summe:	84	83	49	75	77
Prozent:		98,8%	58,3%	89,3%	91,7%

Tabelle 6.2: Litmus ohne ACP: Ergebnisse 2

Test-Auswertung

Die Tabelle 6.3 zeigt die Zusammenfassung der zu erwartenden und der tatsächlich gemessenen Ergebnisse des Konformitätstests ohne Access Control Protocol.

Server	Erwartung	Ergebnisse
Apache2 Linux	hoch 90-100%	100%
Apache2 Win32	hoch 90-100%	100%
Catacomb Linux	hoch 90-100%	100%
Lighttpd Linux	niedrig 70-80%	96,4%
Jakarta Slide Linux	hoch 90-100%	98,8%
Jakarta Slide Win32	hoch 90-100%	98,8%
MS IIS5.1 Win32	mittel 80-90%	58,3%
MS SharePoint 2003 Server	hoch 90-100%	89,3%
Tamino Win32	hoch 90-100%	91,7%

Tabelle 6.3: Litmus ohne ACP: Auswertung

Die Messergebnisse des Konformitätstests weichen teilweise von den erwarteten Ergebnissen ab, was zu überraschen vermag. Während die Server Apache2, Catacomb und Jakarta Slide die erwarteten Testergebnisse bestätigen, weisen die Testreihen der verbleibenden Sever zum Teil große Differenzen in Hinblick auf die Erwartungen auf. So kann Lighttpd eine Konformität vorweisen, welche über den ursprünglichen Erwartungen liegt. Zudem übertreffen die Ergebnisse die bereits auf der Internetpräsenz von Lighthttpd vorliegenden Litmus-Testergebnisse (siehe [33]). Der Microsoft IIS5.1 verhält sich in der Konformität verglichen mit den Lighhttpd Ergebnissen komplementär und kann als *durchgefallen* angesehen werden. Die ermittelte Konformität der beiden kommerziellen Server SharePoint und Tamino liegt unter der erwarteten Konformität. Bezüglich SharePoint lässt sich ableiten, dass die schlechte Konformität gegebenenfalls auf die Verwendung des IIS6.0 zurückzuführen ist, nachdem der IIS5.1 bereits eine äußerst schlechte WebDAV Konformität vorzuweisen hatte. Ähnlich verhält es sich mit Tamino. Dieser setzt ebenfalls auf einem fremden Webserver (hier ein Apache2) auf, und stellt sich als WebDAV Modul für den fremden Webserver dar. Die Integration scheint dabei nicht durchweg optimal gelöst zu sein.

6.2.2 Prestan Performancetest

Die aus den Konformitätstests gewonnenen Ergebnisse sollen für die Festlegung der Test-Erwartungen bei den Performancetests mit einbezogen werden. Der Performancetest wird einmal initiiert und liefert nach Abschluss des Tests die Ergebnisse als eine Auflistung von Methode/Zeit-Wertepaaren zurück.

Test-Erwartung

- **Apache2:** Der Performancetest des Apache2 Servers findet, vergleichbar mit den durchgeführten Konformitätstests, auf zwei unterschiedlichen Plattformen (Windows XP und Ubuntu Linux) statt. Es werden dabei unterschiedliche Testergebnisse erwartet. Auf welcher der beiden Plattformen sich der Apache2 Server performanter verhält, kann auf Grund vieler Faktoren nicht vorhergesagt werden. Da der Apache2 Webserver der meist genutzte Webserver (NetCraft [27]) ist und dieser ständig optimiert [18] wird, wird insgesamt eine sehr hohe Performance erwartet.
- **Catacomb ohne ACP Erweiterung:** Für den Performancetest wird Catacomb ohne ACP Erweiterung verwendet. Catacomb sorgt für eine getrennte Speicherung von Metadaten und den eigentlichen Daten. Während die Metadaten über eine rationale Datenbank (MySQL) verwaltet werden, werden die verwalteten Daten im Dateisystem abgelegt. Dadurch dürften die Testergebnisse in Bezug auf einem unter Linux betriebenen Apache2 Webserver etwas schlechter ausfallen.
- **Lighttpd:** Nach Aussage der Entwickler soll Lighttpd eine höhere Performance aufweisen, als die eines Apache2 Webservers. Diese Aussage wird durch bereits vorliegende Testreihen (Benchmark [30]) belegt. Demnach werden auch im Rahmen der nachfolgend durchgeführten Testreihen bessere Ergebnisse bei den Performancetests erwartet.
- **Jakarta Slide:** Jakarta Slide ist eine auf Java basierende Software und demnach plattformunabhängig. Java stellt eine virtuelle Prozessumgebung, auch als *Java Virtual Machine (JVM)* bekannt, zur Verfügung. Diese ist "nach unten" an das jeweilige Betriebssystem und den jeweiligen Prozessor angepasst. Durch die zusätzliche Zwischenschicht ist davon auszugehen, dass die Ausführungsgeschwindigkeit von Jakarta Slide unter der zusätzlichen Virtualisierung leidet.
- **Microsoft IIS5.1:** Im Kapitel 3 wurde bereits erörtert, dass die Konformität Einfluss auf die Zusammenarbeit zwischen zwei oder mehreren Systemen und demzufolge auch auf die Menge auftretender Verarbeitungs- und Übertragungsfehler hat. Da der Microsoft Internet Information Server 5.1 bei den Konformitätstest das schlechteste Ergebnis erzielt hat, werden auch bei den angestrebten Performancetests die schlechtesten Ergebnisse erwartet.
- **Microsoft SharePoint:** Das gleich wie bei dem Microsoft IIS5.1, gilt auch für den SharePoint Server. Dadurch sollte im Verhältnis zum Kon-

formitätstest die Ergebnisse des Performancetest sein. Die Ergebnisse des Performancetest werden in etwa die der des Jakarta Slide erwartet.

- **Tamino ACP deaktiviert:** Der Tamino WebDAV Server setzt auf den Apache2 Servers auf und bildet die von ihm verwalteten Daten in einer XML-Datenbank ab. Dies ähnelt der Umsetzung von Catacomb, weshalb auf vergleichbare Testergebnisse geschlossen werden kann.

Die Tabelle 6.4 fast die Test-Erwartungen noch einmal zusammen, indem er sie als "Ranking" widerspiegelt.

Rang	Servers
1	Lighttpd Linux
2	Apache2 Win32/Linux
3	Catacomb Linux; Tamino Win32
4	Jakarta Slide Win32/Linux; SharePoint Win32
5	IIS5.1 Win32

Tabelle 6.4: Prestan ohne ACP: Erwartungen

Test-Ergebnisse

Um die Messergebnisse des Performancetests überschaubarer zu machen, werden diese zusammengefasst und in vier Bereiche (props, putget, copymove und locks) eingeteilt. Die Bereiche werden dann nachfolgend als Balkendiagramm abgebildet. Die Einzelergebnisse der jeweiligen Server können dem Anhang dieser Arbeit entnommen werden.

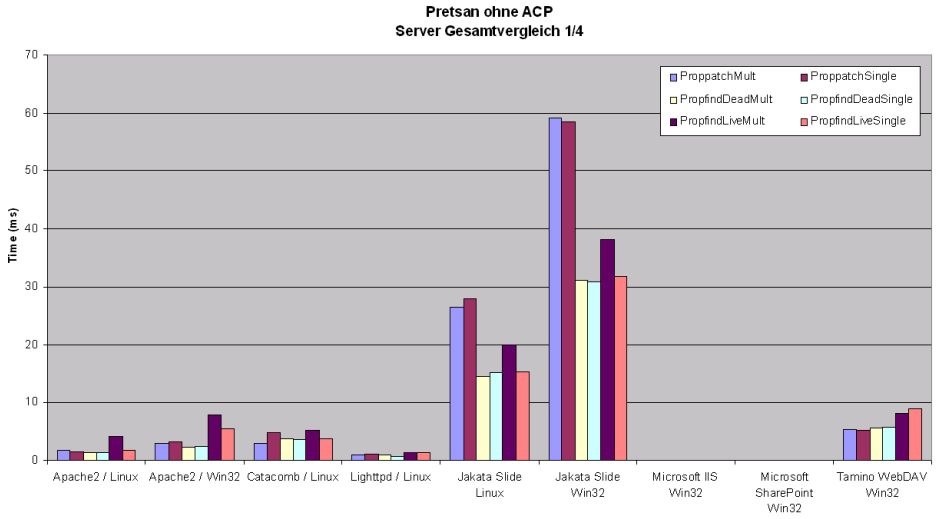

Abbildung 6.3: Prestan ohne ACP: Server Gesamtvergleich 1/4

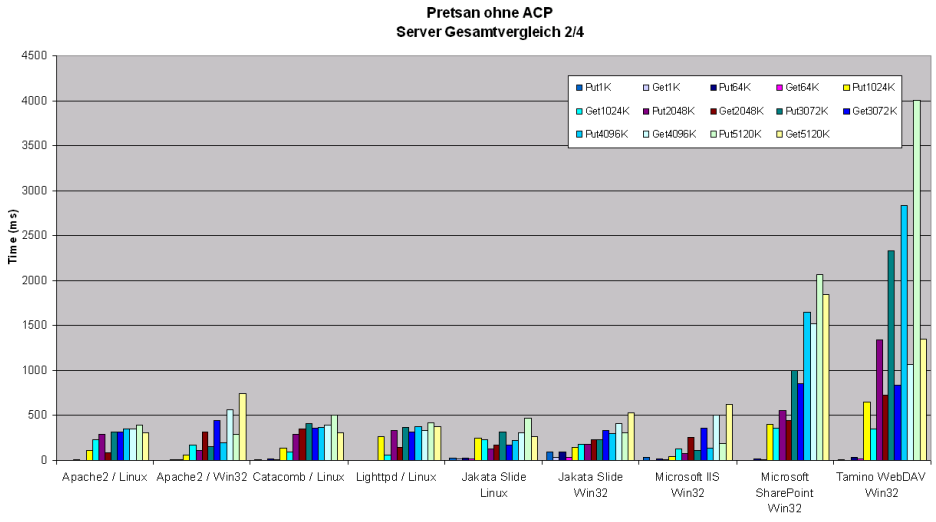

Abbildung 6.4: Prestan ohne ACP: Server Gesamtvergleich 2/4

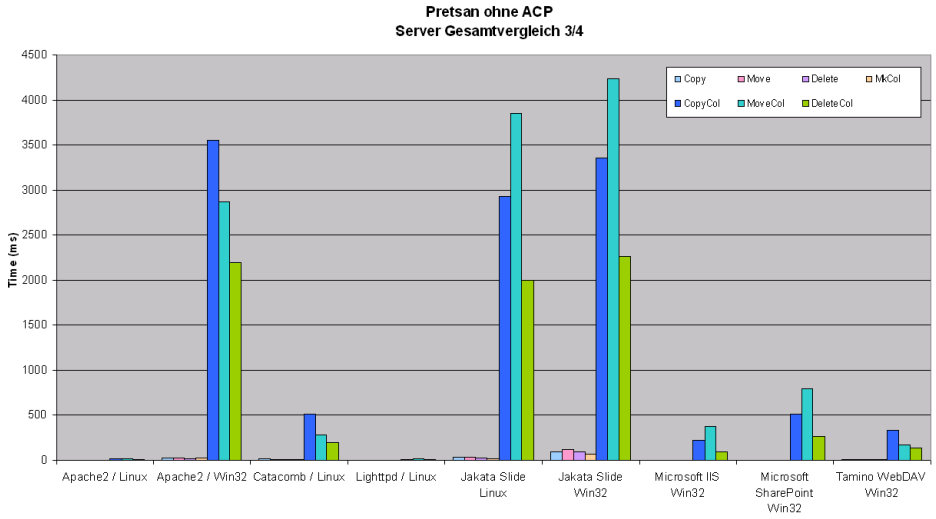

Abbildung 6.5: Prestan ohne ACP: Server Gesamtvergleich 3/4

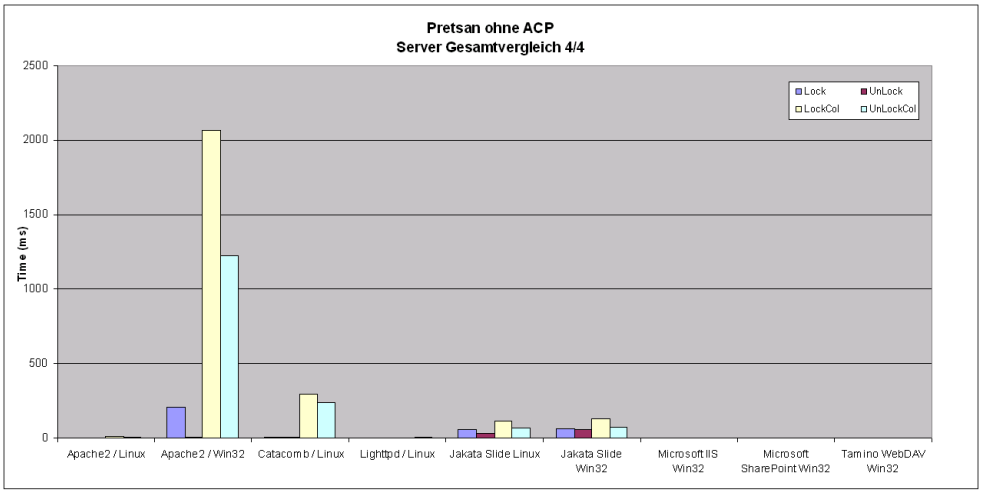

Abbildung 6.6: Prestan ohne ACP: Server Gesamtvergleich 4/4

Test-Auswertung

Um die Ergebnisse bewerten zu können, wird für jeden der vier Bereiche ein Mittelwert über die Messwerte gebildet, bevor die drei besten Bereiche anschließend farblich hervorgehoben werden. Weiterhin werden die Mittelwerte mit Punkten bewertet. Dabei bekommt der beste Wert die höchsten Punkte (neun - Anzahl der Server) und der schlechteste Wert die niedrigsten Punkte (einen).

Folgende Farben visualisieren die ersten drei Ränge des Performancetests ohne ACP:

1. Platz: Grün
2. Platz: Gelb
3. Platz: Rot

Bewertung

Die Tabellen 6.5, 6.6 und 6.7 zeigen die Mittelwerte und die Bewertungen der Messwerte aller Server an.

Server ->	**Apache2 Linux**		**Apache2 Win32**		**Catacomb Linux**	
Methode	**(in ms)**	**(P)**	**(in ms)**	**(P)**	**(in ms)**	**(P)**
props	**2,0**	8	4,1	6	**4,0**	7
putget	**199,3**	7	220,4	5	235,6	3
copymove	**7,5**	8	1248,2	3	149,1	5
locks	**5,2**	8	876,6	4	**65,4**	7
Summe (P)		31		18		22

Tabelle 6.5: Prestan ohne ACP: Bewertung - Ergebnisse 1

Server ->	**Lighttpd Linux**		**Jakarta Slide2.1 Linux**		**Jakarta Slide2.1 Linux**	
Methode	**(in ms)**	**(P)**	**(in ms)**	**(P)**	**(in ms)**	**(P)**
props	**1,1**	9	19,9	4	41,6	3
putget	217,2	6	**190,5**	8	225,5	4
copymove	**5,9**	9	1274,4	2	1464,6	1
locks	**3,1**	9	66,2	6	79,9	5
Summe (P)		33		20		13

Tabelle 6.6: Prestan ohne ACP: Bewertung - Ergebnisse 2

Server -> **Methode**	**Microsoft IIS5.1 Win32** **(in ms)**	**(P)**	**Microsoft SharePoint 2003 Server** **(in ms)**	**(P)**	**Tamino Win32** **(in ms)**	**(P)**
props	-	1	-	1	6,5	5
putget	**181,3**	9	767,4	2	1114,4	1
copymove	100,9	6	226,3	4	**96,0**	7
locks	-	1	-	1	-	1
Summe (P)		17		8		14

Tabelle 6.7: Prestan ohne ACP: Bewertung - Ergebnisse 3

6.2.3 Fazit des Leistungstests ohne ACP

Die Ergebnisse aus den Konformitäts- und den Performancetests ohne ACP haben gezeigt, dass die Open-Source WebDAV Server den kommerziellen WebDAV Servern in vielerlei Hinsicht überlegen sind; sei es bei Güte der Implementierung des WebDAV Standards nach RFC2518 oder bei der Performance, wobei vor allem Letzteres erstaunt.

Um eine Platzierung der WebDAV Server vornehmen zu können, werden die bewerteten Ergebnisse in zwei Kriterien eingeteilt.

- Das erste Kriterium berücksichtigt die Messwerte aller Server, also auch die Server, die zu einer Messung kein Ergebnis liefern konnten.
- Das zweite Kriterium berücksichtigt nur die Messwerte der Server, die zu jeder Messung ein Ergebnis liefern konnten.

Die Tabelle 6.8 zeigt noch einmal die Platzierung der Server an, die sich aus den bewerteten Ergebnissen ergeben haben.

Performancetest ohne ACP Ergebnisse				
	unvollständig		vollständig	
Platz	**Server**	**Punkte**	**Server**	**Punkte**
1	Lighttpd Linux	33	Lighttpd Linux	33
2	Apache2 Linux	31	Apache2 Linux	31
3	Catacomb Linux	22	Catacomb Linux	22
4	Jakarta Slide Linux	20	Jakarta Slide Linux	20
5	Apache2 Win32	18	Apache2 Win32	18
6	IIS5.1 Win32	17	Jakarta Slide Win32	13
7	Tamino Win32	14	IIS5.1 Win32	17
8	Jakarta Slide Win32	13	Tamino Win32	14
9	SharePoint	6	SharePoint	6

Tabelle 6.8: Prestan ohne ACP: Ranking

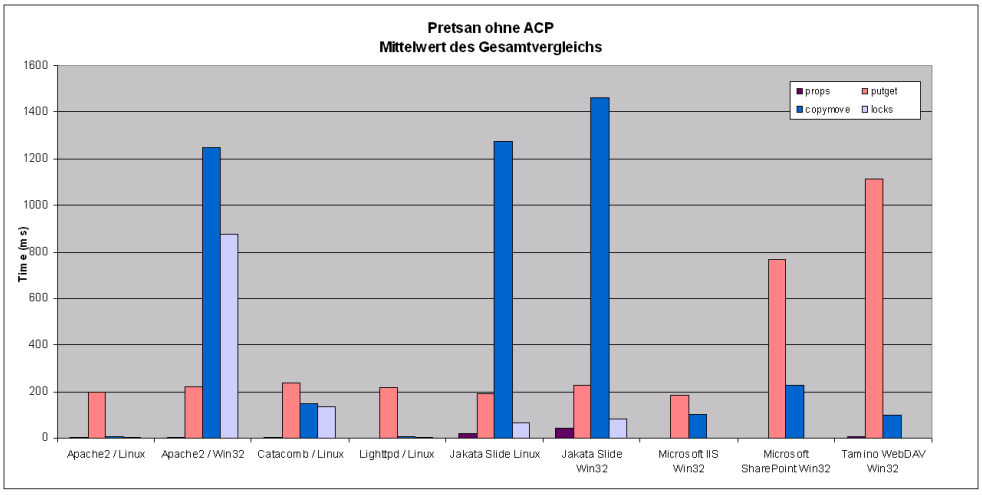

Abbildung 6.7: Prestan ohne ACP: Mittelwerte der Server Gesamtvergleich

6.3 Leistungstest mit Access Control Protocol

Wie schon im vorangegangen Abschnitt, sollen auch im Zuge der Leistungstests mit ACP die zu erwartenden und schlussendlich ermittelten Ergebnisse der Tests dargestellt werden. Dabei fließen alle gewonnen Erkenntnisse aus den vorangegangen Tests ohne ACP mit ein.

6.3.1 Litmus Konformitätstest

Wird ein Konformitätstest mit ACP einem Konformitätstest ohne ACP gegenübergestellt, so ergibt sich bezugnehmend auf die Zusammenstellung der Tests eine Erweiterung der betrachteten fünf Testbereiche um einen weiteren Bereich (acl). Der Test umfasst dabei insgesamt 106 Methoden.

Test-Erwartung

- **Apache2:** Da der Apache2 HTTP Server keine ACP-Unterstützung vorweisen kann, er jedoch bei dem vorangegangenen Konformitätstest 100% erreichen konnte, wird eine mittlere Konformität erwartet.
- **Catacomb mit ACP Erweiterung:** Catacomb unterstützt in seiner neusten Version das ACP. Deshalb wird hier eine hohe Konformität erwartet.
- **Lighttpd:** Lighthttpd kann, vergleichbar mit Apache2, keine ACP-Unterstützung vorweisen. Entsprechend ist lediglich eine mittlere Konformität zum WebDAV-Standard zu erwarten.
- **Jakarta Slide:** Jakarta Slide unterstützt - vergleichbar mit Catacomb - das ACP und konnte in dem vorangegangenen Konformitätstest ohne ACP bereits eine Konformität von 100% vorweisen. Daher wird eine hohe Konformität erwartet.
- **Microsoft IIS5.1:** Da der vorangegangene Konformitätstest des Microsoft Internet Information Server 5.1 ein schlechtes Ergebnis aufzuweisen hatte und zudem keine ACP-Unterstützung existent ist, wird eine sehr niedrige WeBDAV-Konformität erwartet.
- **Microsoft SharePoint:** SharePoint unterstützt ACP. Da möglicherweise nicht alle Funktionalitäten von ACP implementiert sind, wird hier eine mittlere Konformität erwartet.
- **Tamino ACP aktiviert:** Mit der Aktivierung der ACP Erweiterung wird insgesamt eine mittlere Konformität erwartet. Dies entspricht den Ergebnissen des Konformitätstests ohne ACP.

Test-Ergebnisse

In den Tabellen 6.9 und 6.10 werden alle gemessenen Ergebnisse des Konformitätstests mit Access Control Protocol aufgeführt.

Server -> Methode Name/Anzahl		Apache2 Linux	Apache2 Win32	Catacomb Linux	Lighttpd Linux	Jakarta Slide2.1 Linux
basic	14	14	14	14	14	14
copymove	11	11	11	11	11	11
props	25	25	25	25	25	25
locks	30	30	30	30	27	29
http	4	4	4	4	4	4
acl	22	8	8	20	8	21
Summe:	106	92	92	104	98	104
Prozent:		86,8%	86,8%	98,1%	84,0%	98,1%

Tabelle 6.9: Litmus mit ACP: Ergebnisse Teil 1

Server / Methode		Jakarta Win32	Microsoft IIS5.1 Win32	Microsoft SharePoint 2003 Server	Tamino Win32
basic	14	14	14	14	14
copymove	11	11	11	11	11
props	25	25	20	20	24
locks	30	29	-	26	24
http	4	4	4	4	4
acl	22	21	8	17	19
Summe:	106	104	57	92	96
Prozent:		98,1%	53,8%	86,8%	90,6%

Tabelle 6.10: Litmus mit ACP: Ergebnisse Teil 2

Test-Auswertung

Die Tabelle 6.11 zeigt die Zusammenfassung der zu erwartenden und der tatsächlich gemessenen Ergebnisse des Konformitätstests mit Access Control Protocol.

Server	Erwartung	Ergebnisse
Apache2 Linux	mittel 80-90%	86,8%
Apache2 Win32	mittel 80-90%	86,8%
Catacomb Linux	hoch 90-100%	98,1%
Lighttpd Linux	mittel 80-90%	84,0%
Jakarta Slide Linux	hoch 90-100%	98,1%
Jakarta Slide Win32	hoch 90-100%	98,1%
MS IIS5.1 Win32	sehr niedrig $< 70\%$	53,8%
MS SharePoint 2003 Server	mittel 80-90%	86,8%
Tamino WebDAV Win32	mittel 80-90%	90,6%

Tabelle 6.11: Litmus mit ACP: Auswertung

Die gemessenen Ergebnisse des Konformitätstests stimmen mit den erwarteten Ergebnissen überein.

6.3.2 Prestan Performancetests

Der Performancetests mit ACP verlaufen äquivalent zu den Tests ohne ACP.

Test-Erwartung

- **Apache2:** Der Apache2 Server verfügt über keine ACP Erweiterung. Demnach wird er auch keine Ergebnisse bei dem ACP-Test liefern. Auf die anderen Testmethoden hat dies keinen Einfluss, so dass die Ergebnisse dieser Methoden den Ergebnissen des vorangegangen Performancetests entsprechen werden.
- **Catacomb mit ACP Erweiterung:** Für den Performancetest wird Catacomb mit ACP Erweiterung verwendet. Bei der Speicherung der Daten, werden dadurch zusätzliche ACL Informationen in die Datenbank geschrieben. Daraus lässt sich die Vermutung ableiten, dass ein Catacomb mit ACP Erweiterung eine niedrigere Performance vorzuweisen hat, also ein Catacomb ohne ACP Erweiterung.
- **Lighttpd:** Für Lighttpd ist keine ACP Erweiterung verfügbar. Entsprechend werden - im Vergleich zum bereits durchgeführten Performancetest ohne ACP Erweiteung - keine zusätzlichen Testergnisse zur Auswertung zur Verfügung stehen.
- **Jakarta Slide:** Jakarta Slide verfügt über keinerlei Einstellmöglichkeiten der ACP Erweiterung. Deshalb werden zum Performancetest ohne ACP Erweiterung vergleichbare Testergebnisse erwartet.
- **Microsoft IIS5.1:** Der Microsoft Internet Information Server 5.1 verfügt über keine ACP Erweiterung.
- **Microsoft SharePoint:** Für SharePoint werden ähnliche Ergebnisse, wie bei dem Performancetest ohne ACP Erweiterung erwartet.
- **Tamino ACP aktiviert:** Für den Performancetest mit ACP wird die ACP Erweiterung des Tamino Servers aktiviert. So werden auch hier bei der Speicherung der Daten, zusätzliche ACL Informationen in die Datenbank geschrieben.

Die nachfolgende Tabelle fast die Test-Erwartungen noch einmal zusammen und spiegelt diese als "Ranking" wieder. Dabei werden nur die Server berücksichtigt, die dass ACP implementiert haben.

Rang	Servers
1	Catacomb Linux; Tamino Win32
2	Jakarta Slide Linux
3	Jakarta Slide Win32
4	SharePoint Win32

Tabelle 6.12: Prestan mit ACP: Erwartungen

Test-Ergebnisse

Die Messergebnisse dieses Performancetests werden zusammengefasst, in die fünf Bereiche (props, putget, copymove, locks und acl) eingeteilt und nachfolgend als Balkendiagramm abgebildet. Die Einzelergebnisse der jeweiligen Server befinden sich im Anhang.

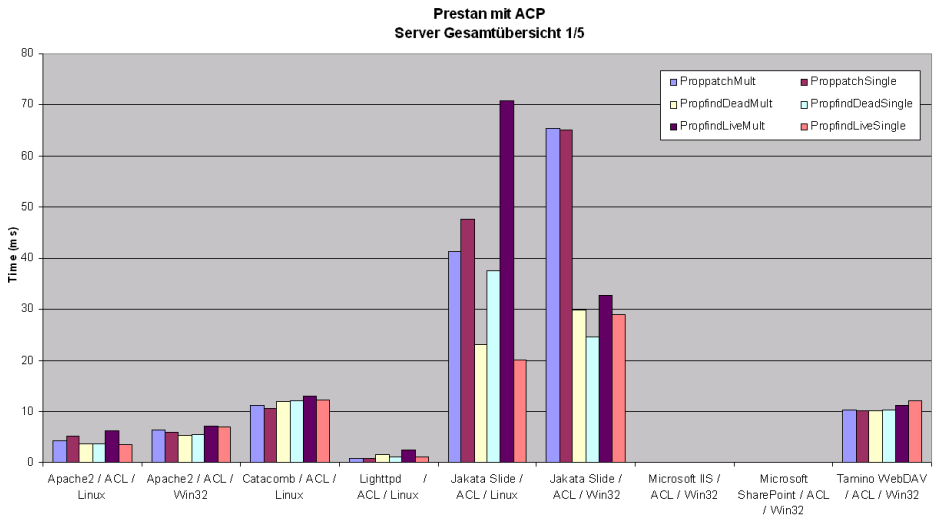

Abbildung 6.8: Prestan mit ACP: Server Gesamtvergleich 1/5

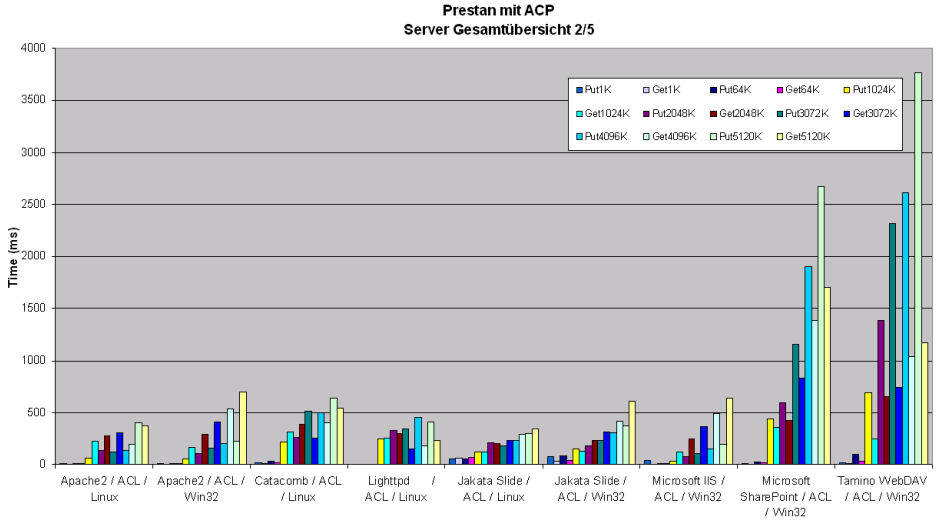

Abbildung 6.9: Prestan mit ACP: Server Gesamtvergleich 2/5

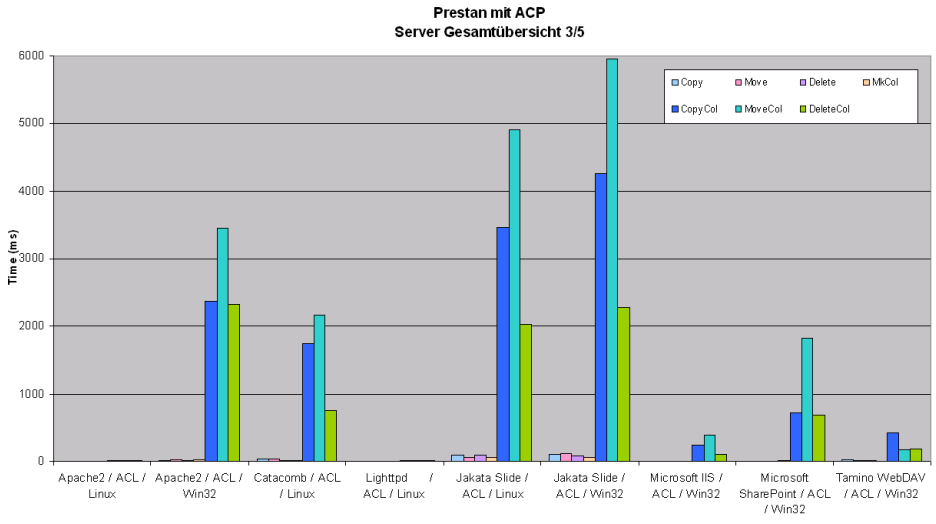

Abbildung 6.10: Prestan mit ACP: Server Gesamtvergleich 3/5

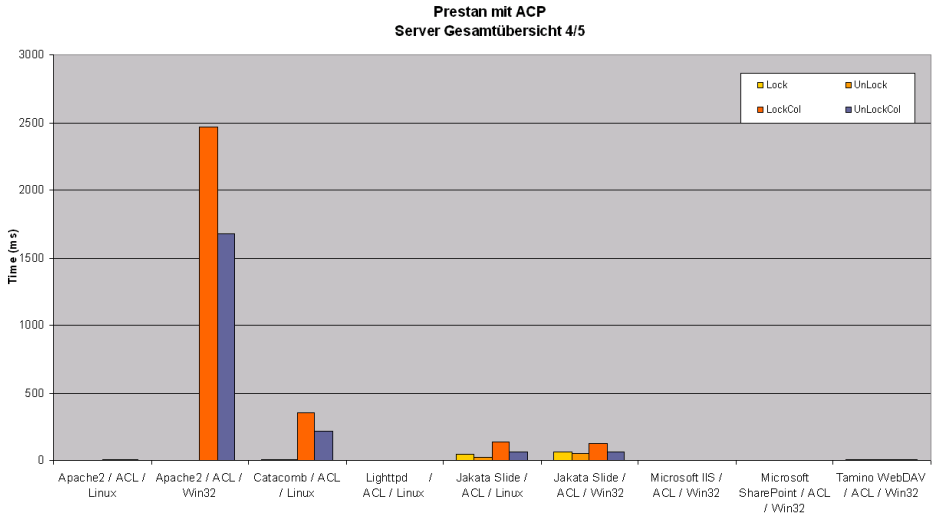

Abbildung 6.11: Prestan mit ACP: Server Gesamtvergleich 4/5

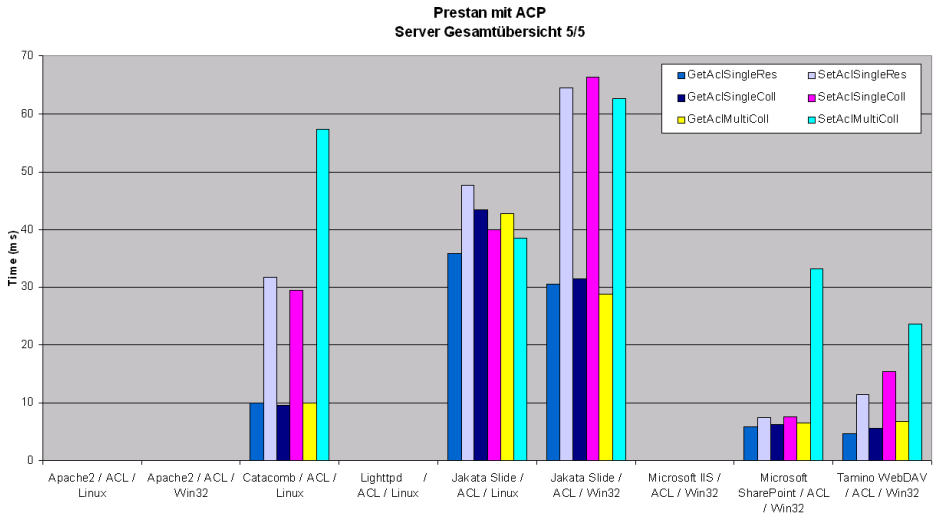

Abbildung 6.12: Prestan mit ACP: Server Gesamtvergleich 5/5

Test-Auswertung

Wie schon bei der Bewertung der Ergebnisse des Performancetests ohne ACP, werden auch hier die dort definierten Kriterien angewendet.

Es wird über jeden der fünf Bereiche ein Mittelwert über die Messwerte gebildet, wobei die drei Bereiche, welche die höchste Bewertung vorzuweisen haben, farblich hervorgehoben werden.

Folgende Farben visualisieren die ersten drei Plätze des Performancetests ohne ACP:

1. Platz: Grün
2. Platz: Gelb
3. Platz: Rot

Bewertung

Die Tabellen 6.13, 6.14 und 6.15 zeigen die Mittelwerte und die Bewertungen der Messwerte aller Server an.

Server ->	**Apache2 Linux**		**Apache2 Win32**		**Catacomb Linux**	
Methode	**(in ms)**	**(P)**	**(in ms)**	**(P)**	**(in ms)**	**(P)**
props	**4,4**	8	**6,2**	7	9,9	6
putget	**164,8**	9	793,8	5	632,7	6
copymove	**7,9**	8	1177,5	3	681,0	4
locks	**4,9**	7	1039,8	3	75,1	5
acl	-	1	-	1	**24,7**	7
Summe (P)		33		19		28

Tabelle 6.13: Prestan mit ACP: Bewertung - Ergebnisse 1

Server ->	**Lighttpd Linux**		**Jakarta Slide2.1 Linux**		**Jakarta Slide2.1 Linux**	
Methode	**(in ms)**	**(P)**	**(in ms)**	**(P)**	**(in ms)**	**(P)**
props	**1,3**	9	40,1	4	41,1	3
putget	**210,8**	8	942,8	4	1148,7	1
copymove	**6,4**	9	1532,3	2	1842,4	1
locks	**2,6**	9	67,5	6	79,9	4
acl	-	1	41,4	6	47,4	5
Summe (P)		36		22		14

Tabelle 6.14: Prestan mit ACP: Bewertung - Ergebnisse 2

Server -> Methode	Microsoft IIS5.1 Win32 (in ms)	(P)	Microsoft SharePoint 2003 Server (in ms)	(P)	Tamino Win32 (in ms)	(P)
props	-	1	-	1	10,7	5
putget	**230,7**	7	1055,5	3	1116,3	2
copymove	**105,6**	7	466,4	5	121,3	6
locks	-	1	-	1	**4,1**	8
acl	-	1	**11,2**	9	**11,3**	8
Summe (P)		17		19		29

Tabelle 6.15: Prestan mit ACP: Bewertung - Ergebnisse 3

6.3.3 Fazit des Leistungstests mit ACP

Werden die Ergebnisse aus den Konformitätstest und den Performancetest mit ACP betrachtet, zeigt sich, dass die Anzahl der WebDAV Server, die das ACP implementiert haben, sehr gering ist. Wird weiterhin die Güte des implementierten WebDAV Standards nach RFC2518 berücksichtigt, so können gerade einmal drei der getesteten Server für eine engere Auswahl vorgesehen werden. Die Open-Source WebDAV Server und die kommerziellen WebDAV Server sind dabei als gleichwertig anzusehen.

Tabelle 6.16 enthält die Platzierungsergebnisse der Server, die aus den bewerteten Ergebnissen gewonnen werden konnten.

Performancetest mit ACP Ergebnisse				
	unvollständig		vollständig	
Platz	Server	Punkte	Server	Punkte
1	Lighttpd Linux	36	Tamino Win32	29
2	Apache2 Linux	33	Catacomb Linux	28
3	Tamino Win32	29	Jakarta Slide Linux	22
4	Catacomb Linux	28	SharePoint	19
5	Jakarta Slide Linux	22	Jakarta Slide Win32	14
6	Apache2 Win32	19	Lighttpd Linux	36
7	SharePoint	19	Apache2 Linux	33
8	IIS5.1 Win32	17	Apache2 Win32	19
9	Jakarta Slide Win32	14	IIS5.1 Win32	17

Tabelle 6.16: Prestan mit ACP: Ranking

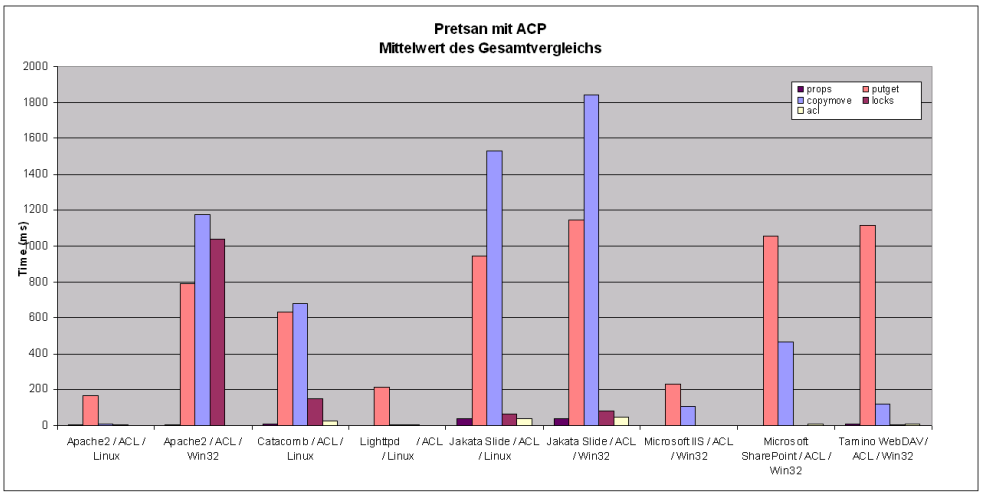

Abbildung 6.13: Prestan mit ACP: Mittelwerte der Server Gesamtvergleich

6.4 Zusammenfassung der Leistungstests

Nachfolgend werden die Ergebnisse der vorangegangenen Leistungstests noch einmal zusammengefasst.

Konformitätstest

Server	Konformitätstest	
	ohne ACP	mit ACP
Apache2 - Linux	100%	86,8%
Apache2 - Win32	100%	86,8%
Catacomb - Linux	100%	98,1%
Lighttpd - Linux	96,4%	84,0%
Jakarta Slide - Linux	98,8%	98,1%
Jakarta Slide - Win32	98,8%	98,1%
MS IIS5.1 - Win32	58,3%	53,8%
MS SharePoint - 2003 Server	89,3%	86,8%
Tamino - Win32	91,7%	90,6%

Tabelle 6.17: Litmus Konformitätstest - Zusammenfassung

Performancetest

Server -> Methode	Apache2 Linux (in ms)	Apache2 Win32 (in ms)	Catacomb Linux (in ms)	Lighttpd Linux (in ms)	Jakata Slide Linux (in ms)	Jakata Slide Win32 (in ms)	MS IIS5.1 Win32 (in ms)	MS SharePoint Win32 (in ms)	Tamino Win32 (in ms)
ProptindDeadSingle	1,5	2,6	3,7	0,9	15,1	30,9	0,0	0,0	5,8
ProptindLiveMult	4,2	7,9	5,2	1,5	20,0	38,1	0,0	0,0	8,1
ProptindLiveSingle	1,8	5,5	3,8	1,4	15,3	31,8	0,0	0,0	9,0
Put1K	2,9	4,2	7,0	1,1	26,6	101,1	39,8	3,3	9,8
Get1K	2,2	2,4	4,0	0,7	17,1	37,5	1,1	2,2	4,4
Put64K	6,1	7,1	20,4	4,0	34,4	99,5	20,6	17,2	39,8
Get64K	4,7	10,1	11,4	0,7	22,8	40,8	9,5	12,3	18,6
Put1024K	109,5	57,1	143,7	274,4	256,5	146,5	50,3	401,2	660,9
Get1024K	230,6	172,7	99,8	64,1	235,0	183,2	133,6	361,9	356,2
Put2048K	288,4	108,5	294,3	330,1	129,7	184,4	78,1	554,7	1346,1
Get2048K	91,3	320,7	357,1	154,0	167,5	236,8	258,6	443,5	726,0
Put3072K	323,5	158,2	415,9	369,5	323,6	236,9	110,0	1001,8	2330,7
Get3072K	325,3	444,5	360,9	325,4	172,9	330,5	366,9	857,6	841,2
Put4096K	353,2	206,5	373,2	386,5	225,4	301,5	146,3	1654,4	2836,1
Get4096K	349,6	560,6	398,4	328,0	308,9	410,9	504,2	1519,7	1074,1
Put5120K	389,0	290,5	501,5	421,5	473,7	316,9	189,5	2065,7	4001,0
Get5120K	314,2	743,0	311,2	380,3	272,6	530,9	629,7	1847,9	1356,2
Copy	2,1	25,7	16,8	1,5	44,2	105,6	0,0	0,0	6,4
Move	2,4	34,1	14,6	0,8	44,3	121,5	0,0	0,0	11,5
Delete	1,5	20,4	6,4	0,9	28,2	96,2	0,0	0,0	6,0
MkCol	1,7	29,1	7,5	0,7	23,6	66,4	1,4	2,8	5,6
CopyCol	17,9	3551,2	512,1	11,9	2933,2	3361,3	218,5	510,7	329,1
MoveCol	15,3	2873,7	285,5	16,5	3853,7	4235,6	382,5	794,0	169,2
DeleteCol	11,6	2203,4	201,0	8,9	1993,9	2265,4	104,1	276,3	144,0
Lock	2,5	205,6	4,3	2,9	53,5	60,6	0,0	0,0	0,0
UnLock	2,4	6,0	4,7	2,5	32,2	57,1	0,0	0,0	0,0
LockCol	9,3	2068,0	125,5	2,8	113,1	129,8	0,0	0,0	0,0
UnLockCol	6,9	1226,7	127,1	4,1	66,0	72,2	0,0	0,0	0,0

Abbildung 6.14: Prestan Performancetest ohne ACP

Server -> Methode	Apache2 ACP / Linux (in ms)	Apache2 ACP / Win32 (in ms)	Catacomb ACP / Linux (in ms)	Lighttpd ACP / Linux (in ms)	Jakata Slide ACP / Linux (in ms)	Jakata Slide ACP / Win32 (in ms)	MS IIS5.1 ACP / Win32 (in ms)	MS SharePoint ACP / Win32 (in ms)	Tamino ACP / Win32 (in ms)
ProppatchMult	4,3	6,5	9,3	1,0	41,3	65,4	0,0	0,0	10,3
ProppatchSingle	5,2	5,9	8,6	0,9	47,7	65,0	0,0	0,0	10,2
PropfindDeadMult	3,7	5,4	9,9	1,5	23,2	29,8	0,0	0,0	10,1
PropfindDeadSingle	3,7	5,4	10,1	1,0	37,5	24,5	0,0	0,0	10,3
PropfindLiveMult	6,3	7,2	10,9	2,5	70,8	32,6	0,0	0,0	11,1
PropfindLiveSingle	3,6	7,0	10,4	1,1	20,0	29,0	0,0	0,0	12,1
Put1K	4,7	5,5	20,5	1,1	51,2	76,5	40,2	4,4	18,1
Get1K	2,4	3,2	12,1	0,7	63,6	30,6	1,1	2,9	4,8
Put64K	4,6	7,5	33,7	3,6	54,1	86,0	6,9	23,2	96,9
Get64K	5,6	9,9	17,5	0,8	66,8	45,8	9,4	15,4	35,3
Put1024K	57,4	56,0	222,2	242,9	123,8	148,4	37,1	435,4	691,4
Get1024K	226,2	164,5	313,3	250,3	124,5	129,3	125,2	356,4	247,2
Put2048K	138,3	107,5	259,0	330,2	208,3	180,7	74,3	595,0	1387,5
Get2048K	277,8	289,6	383,2	294,8	199,0	231,8	247,0	426,7	653,8
Put3072K	124,7	155,6	511,2	341,2	187,5	232,5	108,8	1156,8	2310,1
Get3072K	309,2	411,6	252,1	151,6	233,7	316,4	367,5	829,7	744,9
Put4096K	136,3	200,0	495,2	451,3	238,1	307,2	152,1	1903,8	2614,4
Get4096K	188,1	534,7	399,9	184,7	288,6	416,8	488,4	1392,9	1042,8
Put5120K	400,9	226,5	633,7	412,9	292,9	376,0	195,6	2663,6	3762,4
Get5120K	375,8	697,9	537,2	240,0	340,5	607,2	637,6	1699,3	1169,8
Copy	3,1	17,9	36,0	0,9	92,6	108,4	0,0	0,0	24,4
Move	3,4	32,8	35,8	0,7	67,3	115,3	0,0	0,0	10,3
Delete	2,7	16,2	11,9	0,7	88,0	87,3	0,0	0,0	11,7
MkCol	2,3	31,7	15,3	0,8	63,3	62,7	1,5	9,2	5,9
CopyCol	16,9	2364,0	1744,3	13,8	3467,6	4271,1	242,8	727,9	426,7
MoveCol	15,7	3454,1	2166,9	18,2	4909,8	5962,6	384,8	1835,6	179,5
DeleteCol	11,5	2326,1	756,4	9,8	2037,4	2289,7	109,7	692,0	190,4
Lock	1,7	3,2	9,3	2,9	43,8	67,5	0,0	0,0	4,7
UnLock	1,7	3,0	10,0	2,0	23,8	54,7	0,0	0,0	3,4
LockCol	9,1	2472,5	137,2	2,7	138,3	127,5	0,0	0,0	4,3
UnLockCol	7,0	1680,6	144,0	2,9	64,0	69,9	0,0	0,0	4,0
GetACPSingleRes	0,0	0,0	10,0	0,0	35,9	30,6	0,0	6,0	4,8
SetACPSingleRes	0,0	0,0	31,8	0,0	47,6	64,5	0,0	7,5	11,4
GetACPSingleColl	0,0	0,0	9,6	0,0	43,4	31,5	0,0	6,2	5,6
SetACPSingleColl	0,0	0,0	23,5	0,0	39,9	66,4	0,0	7,7	15,6
GetACPMultiColl	0,0	0,0	10,1	0,0	42,7	28,8	0,0	6,6	6,9
SetACPMultiColl	0,0	0,0	57,4	0,0	38,5	62,6	0,0	33,1	23,6

Abbildung 6.15: Prestan Performancetest mit ACP

Kapitel 7

Zusammenfassung und Ausblicke

7.1 Zusammenfassung

Im Rahmen der hier vorliegenden Diplomarbeit wurde ein Konzept für eine Erweiterung vorliegender WebDAV Test Suites um das in RFC3744 beschriebene Access Control Protocol (ACP) erarbeitet und im Anschluss umgesetzt. Nachfolgend fanden die erweiterten Test Suites Anwendung, als es galt, vorselektierte WebDAV-Implementierungen, welche in Form von WebDAV-Servern vorlagen, auf ihre Konformität zu bestehenden Standards und ihre Leistungsfähigkeit hin zu untersuchen.

Um den Leser beim Verständnis dieser Diplomarbeit unterstützen zu können, wurden zunächst Grundlagen über WebDAV im Allgemeinen, seine Anwendungsmöglichkeiten und das hinter WebDAV stehende Protokoll (inklusive Access Control Protocol) vermittelt. Anschließend wurden zwei - für diese Arbeit wesentliche - Open-Source Projekte vorgestellt, deren Ziel die Schaffung von Möglichkeiten zur Validierung von WebDAV-Implementierungen bezüglich ihrer Konformität und Leistungsfähigkeit darstellt. Vordergründig betrachtet wurden dabei die von den Projekten verfügbar gemachten WebDAV Server Test Suites Litmus und Prestan.

Im Weiteren konnte festgestellt werden, dass die Litmus WebDAV Server Test Suite in ihrer neusten Version bereits eine Access Control Protocol Erweiterung aufzuweisen hat. So konnte in Anlehnung an die bereits existierende Erweiterung, und auf Basis des Software Engineerings, ein Konzept für eine gleichermaßen funktionale Erweiterung bezüglich des Prestan Projektes erarbeitet werden. Im Rahmen der Konzeptfindung und in Hinblick auf die durchzuführenden Testreihen wurde deutlich, dass zusätzlich zur ACP Erweiterung auch eine Erweiterung der gegebenen Testmethoden und der Ergebnisausgabe erfolgen musste, um die Durchführung der Testreihen mit der Prestan Test Suite so effektiv wie möglich zu gestalten.

Der auf den Entwurf des Erweiterungskonzeptes folgenden Implementierung schloss sich eine umfangreiche Validierungsphase der getätigten Implementierung an, bei der es galt, die Korrektheit der Implementierung unter Inhilfenahme des Netzwerk-Protokoll-Analytikers Wireshark zu überprüfen.

Den Abschluss des praktischen Teils dieser Arbeit stellte dann die Verwendung der WebDAV Server Test Suites Litmus und Prestan dar, mit welchen die gewünschten Testreihen an WebDAV Servern, die im Vorfeld durch die Definition eines Kriterienkatalogs aus der Menge verfügbarer Server selektiert worden waren, durchgeführt werden konnten.

7.2 Was bringt die Zukunft

Die Diplomarbeit hat gezeigt, dass zukünftig an den WebDAV Server Test Suite (Litmus und Prestan) und an den Servern noch einiges getan werden kann. Beispielsweise fehlt bei den WebDAV Server Test Suites die Möglichkeit, den Kommunikationsaustausch zum Server mitschneiden zu können, während bezüglich der vorhandenen Menge an WebDAV-Servern ein Mangel an verfügbaren ACP Implementierungen festzustellen.

Weiterhin unterliegen sowohl die Litmus-, als auch die Prestan WebDAV Server TestSuite einigen Einschränkungen, welche es zu beseitigen gilt:

- sind plattformabhängig (Linux),
- stellen sich als Konsolenanwendung dar (ineffiziente Bedienbarkeit),
- bieten eine eingeschränkte Testauswahl an (Konformität und Performance getrennt)

Deshalb sollen nachfolgend einige Vorschläge unterbreitet werden, welche Projekte in der Zukunft umgesetzt werden sollten.

Test Suite Projekte (PHP, Java, C mit NEON Lib)

Die zukünftigen Test Suite Projekte sollten möglichst plattformunabhängig sein. In Frage kommen könnte hierfür eine PHP-Anwendung (über Browser) oder auch eine Java-Anwendung. Diese sind für alle gängigen Plattform verfügbar und portierbar. Außerdem sind sie dadurch bezüglich der Ausgaben nicht mehr auf ein Konsole beschränkt.

Weiterhin sollten die Projekte mögliche Testverfahren vereinen, so dass die Konformität und die Performance unter Einsatz einer einzigen Anwendung getestet werden können.

Server Projekte

Als Serverprojekt ist eine Kombination aus dem performanten Lighttpd und einem WebDAV-Modul mit hoher WebDAV-Konformität, wie Catacomb, denkbar. Ebenfalls interessant erscheint eine Portierung von Catacomb auf Windows-basierende Systeme. Dies würde die Auswahlmöglichkeit an WebDAV Servern, welche eine Unterstützung des Access Control Protocols vorweisen können, steigern.

Literaturverzeichnis

[And02] Andreas Hartl, Franz Prilmeier. *Interdisziplin¨ares Projekt: Anforderungsanalyse*. PDF-Dokument. November 2002

[Dus03] Dusseault, Lisa: *WebDAV – Next-Generation Collaborative Web Authoring*. PRENTICA HALL www.phptr.com,
2003. – ISBN 978–0130652089

[Gie04] Giesecke, Jobst: *Apache HTTP Server – The Apache HTTP Server Documentation Team*. G&U e.Publishing Services GmbH,
2004. – ISBN 978–3826613937

[Kas05] Kastner, Thomas. *Masterarbeit: Document Management in Distributed Project Management Environments*. PDF-Dokument. 2005

[Lit06] Litz, Marcus. *Diplomarbeit: Implementierung des Access Control Protocols in den Catacomb WebDAV Server*. PDF-Dokument. 2006

[PR02] Peter Rossbach, peter Tabatt Andreas Holubek Thomas P.: *Java Servlets und JSP mit Tomcat 4x*. Software und Support Verlag GmbH,
2002. – ISBN 978–3935042277

[Pre04] Pressman, Roger S.: *Software Engineering. A Practitioner's Approach*. McGraw-Hill, Maidenh,
2004. – ISBN 978–0071002325

[Qua98] Quatrain, Terry: *Visual Modeling with Rational Rose and UML*. Addison-Wesley Longman, Amsterdam,
1998. – ISBN 978–0201699616

[Ric07] Richard Stallman. *GNU Coding Standards*. PDF-Dokument. Januar 2007

[Sch06] Schlauch, Tobias. *Masterarbeit: Anbindung von Grid-Speichertechnologien an die Datenmanagement-Software DataFinder*. PDF-Dokument. 2006

Linkverzeichnis

[1] Web-based Distributed Authoring and Versioning (WebDAV)
http://www.webdav.org

[2] DLR Simulations- und Softwaretechnik (SISTEC)
http://www.dlr.de/sc/

[3] Deutsches Zentrum für Luft- und Raumfahrt (DLR)
http://www.dlr.de

[4] Litmus WebDAV Server Compliance Test Suite
http://www.webdav.org/neon/litmus/

[5] Prestan WebDAV Server Performance Test Suite
http://sourceforge.net/projects/prestan/

[6] World Wide Web (WWW)

[7] Internet Engineering Task Force
http://www.ietf.org/

[8] Microsoft FrontPage

[9] Adobe
www.adobe.com

[10] Cobra Computer's Brainware
http://www.cobra.de/

[11] Hypertext Transfer Protocol (HTTP)
http://de.wikipedia.org/wiki/Http

[12] Davfs2 - WebDAV Client
http://dav.sourceforge.net/

[13] Cadaver Command-Lline WebDAV Client
http://www.webdav.org/cadaver/

[14] Sitecopy - WebDAV Client
http://www.lyra.org/sitecopy/

[15] Goliath - WebDAV Client
http://www.webdav.org/goliath/

[16] Webdrive - WebDAV Client
www.webdrive.com/

[17] DAV Explorer - WebDAV Client
http://www.davexplorer.org/

[18] Apache HTTP Server Project
http://httpd.apache.org/

[19] Jakarta Slide Projekt
http://jakarta.apache.org/slide/index.html

[20] Zope Community - WebDAV Server
http://www.zope.org/

[21] Lighttpd HTTP Server
www.lighttpd.net/

[22] Microsoft Internet Information Server
http://www.microsoft.com

[23] Tamino XML - Webdav Server
http://www.softwareag.com/de/products/tamino/

[24] Institute of Electrical and Electronics Engineers (IEEE)
http://www.ieee.org/

[25] IEEE Std 610.12-1990 Definitions for Software Engineering Terms
http://standards.ieee.org

[26] Wireshark Network Protocol Analyzer
http://www.wireshark.org/

[27] Netcraft Service Outage for Fasthosts
http://www.netcraft.com/

[28] Apache HTTP Server

[29] Catacomb WebDAV Repository Modul
http://catacomb.tigris.org/

[30] Lighttpd Benchmark
http://www.lighttpd.net/benchmark

[31] Microsoft Office SharePoint Server
http://office.microsoft.com

[32] Apache Modul mod_dav
http://www.webdav.org/mod_dav/

[33] Lighttpd Modul WebDAV
http://trac.lighttpd.net/trac/wiki/

RFC-Verzeichnis

[RFC1866] Berners-Lee, T. ; Connolly, D.: *Hypertext Markup Language - 2.0*. RFC 1866 (Historic). November 1995 (Request for Comments). – Obsoleted by RFC 2854

[RFC3986] Berners-Lee, T. ; Fielding, R. ; Masinter, L. *Uniform Resource Identifier (URI): Generic Syntax*. RFC 3986 (Standard). Januar 2005

[RFC3253] Clemm, G. ; Amsden, J. ; Ellison, T. ; Kaler, C. ; Whitehead, J. *Versioning Extensions to WebDAV (Web Distributed Authoring and Versioning)*. RFC 3253 (Proposed Standard). März 2002

[RFC3744] Clemm, G. ; Reschke, J. ; Sedlar, E. ; Whitehead, J. *Web Distributed Authoring and Versioning (WebDAV) Access Control Protocol*. RFC 3744 (Proposed Standard). Mai 2004

[RFC2068] Fielding, R. ; Gettys, J. ; Mogul, J. ; Frystyk, H. ; Berners-Lee, T.: *Hypertext Transfer Protocol – HTTP/1.1*. RFC 2068 (Proposed Standard). Januar 1997 (Request for Comments). – Obsoleted by RFC 2616

[RFC2616] Fielding, R. ; Gettys, J. ; Mogul, J. ; Frystyk, H. ; Masinter, L. ; Leach, P. ; Berners-Lee, T.: *Hypertext Transfer Protocol – HTTP/1.1*. RFC 2616 (Draft Standard). Juni 1999 (Request for Comments). – Updated by RFC 2817

[RFC2518] Goland, Y. ; Whitehead, E. ; Faizi, A. ; Carter, S. ; Jensen, D. *HTTP Extensions for Distributed Authoring – WEBDAV*. RFC 2518 (Proposed Standard). Februar 1999

[RFC1739] Kessler, G. ; Shepard, S.: *A Primer On Internet and TCP/IP Tools*. RFC 1739 (Informational). Dezember 1994 (Request for Comments). – Obsoleted by RFC 2151

[RFC793] Postel, J.: *Transmission Control Protocol*. RFC 793 (Standard). September 1981 (Request for Comments). – Updated by RFC 3168

[RFC959] Postel, J. ; Reynolds, J.: *File Transfer Protocol*. RFC 959 (Standard). Oktober 1985 (Request for Comments). – Updated by RFCs 2228, 2640, 2773, 3659

[RFC2291] Slein, J. ; Vitali, F. ; Whitehead, E. ; Durand, D. *Requirements for a Distributed Authoring and Versioning Protocol for the World Wide Web*. RFC 2291 (Informational). Februar 1998

Anhang A

Litmus Testergebnisse

A.1 Litmus ohne ACP

```
ubuntu@ubuntuserver:~# litmus http://192.168.27.130/webdav webdav webdav
-> running `basic':
 0. init.................. pass
 1. begin................. pass
 2. options............... pass
 3. put_get............... pass
 4. put_get_utf8_segment.. pass
 5. mkcol_over_plain...... pass
 6. delete................ pass
 7. delete_null........... pass
 8. mkcol................. pass
 9. mkcol_again........... pass
10. delete_coll........... pass
11. mkcol_no_parent....... pass
12. mkcol_with_body....... pass
13. finish................ pass
<- summary for `basic': of 14 tests run: 14 passed, 0 failed. 100.0%
-> running `copymove':
 0. init.................. pass
 1. begin................. pass
 2. copy_init............. pass
 3. copy_simple........... pass
 4. copy_overwrite........ pass
 5. copy_cleanup.......... pass
 6. copy_coll............. pass
 7. move.................. pass
 8. move_coll............. pass
 9. move_cleanup.......... pass
10. finish................ pass
<- summary for `copymove': of 11 tests run: 11 passed, 0 failed. 100.0%
-> running `props':
 0. init.................. pass
 1. begin................. pass
 2. propfind_invalid...... pass
 3. propfind_invalid2..... pass
 4. propfind_d0........... pass
 5. propinit.............. pass
 6. propset............... pass
 7. propget............... pass
 8. propmove.............. pass
 9. propget............... pass
10. propdeletes........... pass
11. propget............... pass
12. propreplace........... pass
13. propget............... pass
14. propnullns............ pass
15. propget............... pass
16. prophighunicode....... pass
17. propget............... pass
18. propvalnspace......... pass
19. propwformed........... pass
20. propinit.............. pass
21. propmanyns............ pass
22. propget............... pass
23. propcleanup........... pass
24. finish................ pass
<- summary for `props': of 25 tests run: 25 passed, 0 failed. 100.0%
```

```
-> running `locks':
 0. init.................. pass
 1. begin................. pass
 2. options............... pass
 3. precond............... pass
 4. init_locks............ pass
 5. put................... pass
 6. lock_excl............. pass
 7. discover.............. pass
 8. notowner_modify....... pass
 9. notowner_lock......... pass
10. owner_modify.......... pass
11. notowner_modify....... pass
12. notowner_lock......... pass
13. copy.................. pass
14. cond_put.............. pass
15. fail_cond_put......... pass
16. cond_put_with_not..... pass
17. cond_put_corrupt_token WARNING: PUT failed with 400 not 423
    ..................... pass (with 1 warning)
18. complex_cond_put...... pass
19. fail_complex_cond_put. pass
20. unlock................ pass
21. lock_shared........... pass
22. notowner_modify....... pass
23. notowner_lock......... pass
24. owner_modify.......... pass
25. double_sharedlock..... pass
26. notowner_modify....... pass
27. notowner_lock......... pass
28. unlock................ pass
29. finish................ pass
<- summary for `locks': of 30 tests run: 30 passed, 0 failed. 100.0%
-> 1 warning was issued.
-> running `http':
 0. init.................. pass
 1. begin................. pass
 2. expect100............. pass
 3. finish................ pass
<- summary for `http': of 4 tests run: 4 passed, 0 failed. 100.0%
ubuntu@ubuntuserver:~#

.
```

Listing A.1: Litmus ohne ACP: Apache2 Linux

```
ubuntu@ubuntuserver:~# litmus http://192.168.27.132/webdav webdav webdav
-> running `basic':
 0. init.................. pass
 1. begin................. pass
 2. options............... pass
 3. put_get............... pass
 4. put_get_utf8_segment.. pass
 5. mkcol_over_plain...... pass
 6. delete................ pass
 7. delete_null........... pass
 8. mkcol................. pass
 9. mkcol_again........... pass
10. delete_coll........... pass
11. mkcol_no_parent....... pass
12. mkcol_with_body....... pass
13. finish................ pass
<- summary for `basic': of 14 tests run: 14 passed, 0 failed. 100.0%
-> running `copymove':
 0. init.................. pass
 1. begin................. pass
 2. copy_init............. pass
 3. copy_simple........... pass
 4. copy_overwrite........ pass
 5. copy_cleanup.......... pass
 6. copy_coll............. pass
 7. move.................. pass
 8. move_coll............. pass
 9. move_cleanup.......... pass
10. finish................ pass
<- summary for `copymove': of 11 tests run: 11 passed, 0 failed. 100.0%
-> running `props':
 0. init.................. pass
 1. begin................. pass
 2. propfind_invalid...... pass
 3. propfind_invalid2..... pass
 4. propfind_d0........... pass
 5. propinit.............. pass
 6. propset............... pass
```

```
 7. propget ..............  pass
 8. propmove .............  pass
 9. propget ..............  pass
10. propdeletes ..........  pass
11. propget ..............  pass
12. propreplace ..........  pass
13. propget ..............  pass
14. propnullns ...........  pass
15. propget ..............  pass
16. prophighunicode ......  pass
17. propget ..............  pass
18. propvalnspace ........  pass
19. propwformed ..........  pass
20. propinit .............  pass
21. propmanyns ...........  pass
22. propget ..............  pass
23. propcleanup ..........  pass
24. finish ...............  pass
<- summary for 'props': of 25 tests run: 25 passed, 0 failed. 100.0%
-> running 'locks':
 0. init ..................  pass
 1. begin .................  pass
 2. options ...............  pass
 3. precond ...............  pass
 4. init_locks ............  pass
 5. put ...................  pass
 6. lock_excl .............  pass
 7. discover ..............  pass
 8. notowner_modify .......  pass
 9. notowner_lock .........  pass
10. owner_modify ..........  pass
11. notowner_modify .......  pass
12. notowner_lock .........  pass
13. copy ..................  pass
14. cond_put ..............  pass
15. fail_cond_put .........  pass
16. cond_put_with_not .....  pass
17. cond_put_corrupt_token WARNING: PUT failed with 400 not 423
    ......................  pass (with 1 warning)
18. complex_cond_put ......  pass
19. fail_complex_cond_put.  pass
20. unlock ................  pass
21. lock_shared ...........  pass
22. notowner_modify .......  pass
23. notowner_lock .........  pass
24. owner_modify ..........  pass
25. double_sharedlock .....  pass
26. notowner_modify .......  pass
27. notowner_lock .........  pass
28. unlock ................  pass
29. finish ................  pass
<- summary for 'locks': of 30 tests run: 30 passed, 0 failed. 100.0%
-> 1 warning was issued.
-> running 'http':
 0. init ..................  pass
 1. begin .................  pass
 2. expect100 .............  pass
 3. finish ................  pass
<- summary for 'http': of 4 tests run: 4 passed, 0 failed. 100.0%
ubuntu@ubuntuserver:~#

.
```

Listing A.2: Litmus ohne ACP: Apache2 Win32

```
ubuntu@ubuntuserver:~# litmus http://192.168.27.135/dfCatacomb datafinder 123
-> running 'basic':
 0. init ..................  pass
 1. begin .................  pass
 2. options ...............  pass
 3. put_get ...............  pass
 4. put_get_utf8_segment..  pass
 5. mkcol_over_plain ......  pass
 6. delete ................  pass
 7. delete_null ...........  pass
 8. mkcol .................  pass
 9. mkcol_again ...........  pass
10. delete_coll ...........  pass
11. mkcol_no_parent .......  pass
12. mkcol_with_body .......  pass
13. finish ................  pass
<- summary for 'basic': of 14 tests run: 14 passed, 0 failed. 100.0%
-> running 'copymove':
 0. init ..................  pass
```

```
 1. begin.................. pass
 2. copy_init.............. pass
 3. copy_simple............ pass
 4. copy_overwrite........ pass
 5. copy_cleanup.......... pass
 6. copy_coll.............. pass
 7. move................... pass
 8. move_coll.............. pass
 9. move_cleanup........... pass
10. finish................. pass
<- summary for 'copymove': of 11 tests run: 11 passed, 0 failed. 100.0%
-> running 'props':
 0. init................... pass
 1. begin.................. pass
 2. propfind_invalid....... pass
 3. propfind_invalid2..... pass
 4. propfind_d0............ pass
 5. propinit............... pass
 6. propset................ pass
 7. propget................ pass
 8. propmove............... pass
 9. propget................ pass
10. propdeletes............ pass
11. propget................ pass
12. propreplace............ pass
13. propget................ pass
14. propnullns............. pass
15. propget................ pass
16. prophighunicode....... pass
17. propget................ pass
18. propvalnspace......... pass
19. propwformed........... pass
20. propinit............... pass
21. propmanyns............. pass
22. propget................ pass
23. propcleanup........... pass
24. finish................. pass
<- summary for 'props': of 25 tests run: 25 passed, 0 failed. 100.0%
-> running 'locks':
 0. init................... pass
 1. begin.................. pass
 2. options................ pass
 3. precond................ pass
 4. init_locks............. pass
 5. put.................... pass
 6. lock_excl.............. pass
 7. discover............... pass
 8. notowner_modify....... pass
 9. notowner_lock......... pass
10. owner_modify.......... pass
11. notowner_modify....... pass
12. notowner_lock......... pass
13. copy................... pass
14. cond_put............... pass
15. fail_cond_put......... WARNING: PUT failed with 423 not 412
    ....................... pass (with 1 warning)
16. cond_put_with_not..... pass
17. cond_put_corrupt_token WARNING: PUT failed with 400 not 423
    ....................... pass (with 1 warning)
18. complex_cond_put...... pass
19. fail_complex_cond_put. pass
20. unlock................ pass
21. lock_shared........... pass
22. notowner_modify....... pass
23. notowner_lock......... pass
24. owner_modify.......... pass
25. double_sharedlock..... pass
26. notowner_modify....... pass
27. notowner_lock......... pass
28. unlock................ pass
29. finish................. pass
<- summary for 'locks': of 30 tests run: 30 passed, 0 failed. 100.0%
-> 2 warnings were issued.
-> running 'http':
 0. init................... pass
 1. begin.................. pass
 2. expect100.............. pass
 3. finish................. pass
<- summary for 'http': of 4 tests run: 4 passed, 0 failed. 100.0%
ubuntu@ubuntuserver:~#

.
```

Listing A.3: Litmus ohne ACP: Catacomb ohne ACP Linux

```
ubuntu@ubuntuserver:~# litmus http://192.168.239.144/dav root root
-> running `basic':
 0. init.................. pass
 1. begin................. pass
 2. options............... pass
 3. put_get............... pass
 4. put_get_utf8_segment.. pass
 5. mkcol_over_plain...... pass
 6. delete................ pass
 7. delete_null........... pass
 8. mkcol................. pass
 9. mkcol_again........... pass
10. delete_coll........... pass
11. mkcol_no_parent....... pass
12. mkcol_with_body....... pass
13. finish................ pass
<- summary for `basic': of 14 tests run: 14 passed, 0 failed. 100.0%
-> running `copymove':
 0. init.................. pass
 1. begin................. pass
 2. copy_init............. pass
 3. copy_simple........... pass
 4. copy_overwrite........ WARNING:
 COPY to existing resource didn't give 204
    ....................... pass (with 1 warning)
 5. copy_cleanup.......... pass
 6. copy_coll............. pass
 7. move.................. WARNING:
 MOVE to existing collection resource didn't give 204
    ....................... pass (with 1 warning)
 8. move_coll............. pass
 9. move_cleanup.......... pass
10. finish................ pass
<- summary for `copymove': of 11 tests run: 11 passed, 0 failed. 100.0%
-> 2 warnings were issued.
-> running `props':
 0. init.................. pass
 1. begin................. pass
 2. propfind_invalid...... pass
 3. propfind_invalid2..... pass
 4. propfind_d0........... pass
 5. propinit.............. pass
 6. propset............... pass
 7. propget............... pass
 8. propmove.............. pass
 9. propget............... pass
10. propdeletes........... pass
11. propget............... pass
12. propreplace........... pass
13. propget............... pass
14. propnullns............ pass
15. propget............... pass
16. prophighunicode....... pass
17. propget............... pass
18. propvalnspace......... pass
19. propwformed........... pass
20. propinit.............. pass
21. propmanyns............ pass
22. propget............... pass
23. propcleanup........... pass
24. finish................ pass
<- summary for `props': of 25 tests run: 25 passed, 0 failed. 100.0%
-> running `locks':
 0. init.................. pass
 1. begin................. pass
 2. options............... pass
 3. precond............... pass
 4. init_locks............ pass
 5. put................... pass
 6. lock_excl............. pass
 7. discover.............. pass
 8. notowner_modify....... pass
 9. notowner_lock......... pass
10. owner_modify.......... pass
11. notowner_modify....... pass
12. notowner_lock......... pass
13. copy.................. pass
14. cond_put.............. pass
15. fail_cond_put......... FAIL
(conditional PUT with invalid lock-token should fail: 200 OK)
16. cond_put_with_not..... pass
17. cond_put_corrupt_token WARNING:
PUT failed with 200 not 423
    ....................... pass (with 1 warning)
18. complex_cond_put...... pass
19. fail_complex_cond_put. FAIL
```

```
(PUT with complex bogus conditional should fail with 412: 200 OK)
20. unlock.................. pass
21. lock_shared............. pass
22. notowner_modify........ pass
23. notowner_lock.......... pass
24. owner_modify........... pass
25. double_sharedlock..... FAIL (shared LOCK on locked resource:
423 Locked)
26. notowner_modify........ pass
27. notowner_lock.......... pass
28. unlock.................. pass
29. finish.................. pass
<- summary for 'locks': of 30 tests run: 27 passed, 3 failed. 90.0%
-> 1 warning was issued.
-> running 'http':
 0. init.................... pass
 1. begin................... pass
 2. expect100............... pass
 3. finish.................. pass
<- summary for 'http': of 4 tests run: 4 passed, 0 failed. 100.0%
See debug.log for network/debug traces.
ubuntu@ubuntuserver:~#

.
```

Listing A.4: Litmus ohne ACP: Lighttpd Linux

```
ubuntu@ubuntuserver:/var/log# litmus http://192.168.27.139:8080/slide/files root root
-> running 'basic':
 0. init.................... pass
 1. begin................... pass
 2. options................. pass
 3. put_get................. pass
 4. put_get_utf8_segment.. pass
 5. mkcol_over_plain...... pass
 6. delete.................. pass
 7. delete_null............. pass
 8. mkcol................... pass
 9. mkcol_again............. pass
10. delete_coll............. pass
11. mkcol_no_parent........ pass
12. mkcol_with_body........ pass
13. finish.................. pass
<- summary for 'basic': of 14 tests run: 14 passed, 0 failed. 100.0%
-> running 'copymove':
 0. init.................... pass
 1. begin................... pass
 2. copy_init............... pass
 3. copy_simple............. pass
 4. copy_overwrite.......... pass
 5. copy_cleanup............ pass
 6. copy_coll............... pass
 7. move.................... pass
 8. move_coll............... pass
 9. move_cleanup............ pass
10. finish.................. pass
<- summary for 'copymove': of 11 tests run: 11 passed, 0 failed. 100.0%
-> running 'props':
 0. init.................... pass
 1. begin................... pass
 2. propfind_invalid....... pass
 3. propfind_invalid2...... pass
 4. propfind_d0............. pass
 5. propinit................ pass
 6. propset................. pass
 7. propget................. pass
 8. propmove................ pass
 9. propget................. pass
10. propdeletes............. pass
11. propget................. pass
12. propreplace............. pass
13. propget................. pass
14. propnullns.............. pass
15. propget................. pass
16. prophighunicode........ pass
17. propget................. pass
18. propvalnspace.......... pass
19. propwformed............ pass
20. propinit................ pass
21. propmanyns.............. pass
22. propget................. pass
23. propcleanup............. pass
24. finish.................. pass
<- summary for 'props': of 25 tests run: 25 passed, 0 failed. 100.0%
```

```
-> running `locks':
 0. init.................. pass
 1. begin................. pass
 2. options............... pass
 3. precond............... pass
 4. init_locks............ pass
 5. put................... pass
 6. lock_excl............. pass
 7. discover.............. pass
 8. notowner_modify....... pass
 9. notowner_lock......... pass
10. owner_modify.......... pass
11. notowner_modify....... pass
12. notowner_lock......... pass
13. copy.................. pass
14. cond_put.............. pass
15. fail_cond_put......... WARNING:
PUT failed with 423 not 412
    ....................... pass (with 1 warning)
16. cond_put_with_not..... pass
17. cond_put_corrupt_token pass
18. complex_cond_put...... pass
19. fail_complex_cond_put. FAIL
(PUT with complex bogus conditional should fail with 412: 204 No Content)
20. unlock................ pass
21. lock_shared........... pass
22. notowner_modify....... pass
23. notowner_lock......... pass
24. owner_modify.......... pass
25. double_sharedlock..... pass
26. notowner_modify....... pass
27. notowner_lock......... pass
28. unlock................ pass
29. finish................ pass
<- summary for `locks': of 30 tests run: 29 passed, 1 failed. 96.7%
-> 1 warning was issued.
-> running `http':
 0. init.................. pass
 1. begin................. pass
 2. expect100............. pass
 3. finish................ pass
<- summary for `http': of 4 tests run: 4 passed, 0 failed. 100.0%
See debug.log for network/debug traces.
ubuntu@ubuntuserver:/var/log#

.
```

Listing A.5: Litmus ohne ACP: Jakarta Slide2.1 Linux

```
ubuntu@ubuntuserver:/var/log# litmus http://192.168.27.138:8080/slide/files root root
-> running `basic':
 0. init.................. pass
 1. begin................. pass
 2. options............... pass
 3. put_get............... pass
 4. put_get_utf8_segment.. pass
 5. mkcol_over_plain...... pass
 6. delete................ pass
 7. delete_null........... pass
 8. mkcol................. pass
 9. mkcol_again........... pass
10. delete_coll........... pass
11. mkcol_no_parent....... pass
12. mkcol_with_body....... pass
13. finish................ pass
<- summary for `basic': of 14 tests run: 14 passed, 0 failed. 100.0%
-> running `copymove':
 0. init.................. pass
 1. begin................. pass
 2. copy_init............. pass
 3. copy_simple........... pass
 4. copy_overwrite........ pass
 5. copy_cleanup.......... pass
 6. copy_coll............. pass
 7. move.................. pass
 8. move_coll............. pass
 9. move_cleanup.......... pass
10. finish................ pass
<- summary for `copymove': of 11 tests run: 11 passed, 0 failed. 100.0%
-> running `props':
 0. init.................. pass
 1. begin................. pass
 2. propfind_invalid...... pass
 3. propfind_invalid2..... pass
```

```
 4. propfind_d0 ..... pass
 5. propinit ..... pass
 6. propset ..... pass
 7. propget ..... pass
 8. propmove ..... pass
 9. propget ..... pass
10. propdeletes ..... pass
11. propget ..... pass
12. propreplace ..... pass
13. propget ..... pass
14. propnullns ..... pass
15. propget ..... pass
16. prophighunicode ..... pass
17. propget ..... pass
18. propvalnspace ..... pass
19. propwformed ..... pass
20. propinit ..... pass
21. propmanyns ..... pass
22. propget ..... pass
23. propcleanup ..... pass
24. finish ..... pass
<- summary for 'props': of 25 tests run: 25 passed, 0 failed. 100.0%
-> running 'locks':
 0. init ..... pass
 1. begin ..... pass
 2. options ..... pass
 3. precond ..... pass
 4. init_locks ..... pass
 5. put ..... pass
 6. lock_excl ..... pass
 7. discover ..... pass
 8. notowner_modify ..... pass
 9. notowner_lock ..... pass
10. owner_modify ..... pass
11. notowner_modify ..... pass
12. notowner_lock ..... pass
13. copy ..... pass
14. cond_put ..... pass
15. fail_cond_put ..... WARNING:
PUT failed with 423 not 412
     ..... pass (with 1 warning)
16. cond_put_with_not ..... pass
17. cond_put_corrupt_token pass
18. complex_cond_put ..... pass
19. fail_complex_cond_put. FAIL
(PUT with complex bogus conditional should fail with 412: 204 No Content)
20. unlock ..... pass
21. lock_shared ..... pass
22. notowner_modify ..... pass
23. notowner_lock ..... pass
24. owner_modify ..... pass
25. double_sharedlock ..... pass
26. notowner_modify ..... pass
27. notowner_lock ..... pass
28. unlock ..... pass
29. finish ..... pass
<- summary for 'locks': of 30 tests run: 29 passed, 1 failed. 96.7%
-> 1 warning was issued.
-> running 'http':
 0. init ..... pass
 1. begin ..... pass
 2. expect100 ..... pass
 3. finish ..... pass
<- summary for 'http': of 4 tests run: 4 passed, 0 failed. 100.0%
See debug.log for network/debug traces.
ubuntu@ubuntuserver:/var/log#

.
```

Listing A.6: Litmus ohne ACP: Jakarta Slide2.1 Win32

```
ubuntu@ubuntuserver:~# litmus http://192.168.27.137/webdav webdavuser webdav
-> running 'basic':
 0. init ..... pass
 1. begin ..... pass
 2. options ..... pass
 3. put_get ..... pass
 4. put_get_utf8_segment.. pass
 5. mkcol_over_plain ..... pass
 6. delete ..... pass
 7. delete_null ..... pass
 8. mkcol ..... pass
 9. mkcol_again ..... pass
10. delete_coll ..... pass
```

```
11. mkcol_no_parent ..... pass
12. mkcol_with_body ..... pass
13. finish ..... pass
<- summary for 'basic': of 14 tests run: 14 passed, 0 failed. 100.0%
-> running 'copymove':
 0. init ..... pass
 1. begin ..... pass
 2. copy_init ..... pass
 3. copy_simple ..... pass
 4. copy_overwrite ..... pass
 5. copy_cleanup ..... pass
 6. copy_coll ..... pass
 7. move ..... pass
 8. move_coll ..... pass
 9. move_cleanup ..... pass
10. finish ..... pass
<- summary for 'copymove': of 11 tests run: 11 passed, 0 failed. 100.0%
-> running 'props':
 0. init ..... pass
 1. begin ..... pass
 2. propfind_invalid ..... pass
 3. propfind_invalid2 ..... pass
 4. propfind_d0 ..... pass
 5. propinit ..... pass
 6. propset ..... pass
 7. propget ..... pass
 8. propmove ..... pass
 9. propget ..... pass
10. propdeletes ..... pass
11. propget ..... pass
12. propreplace ..... pass
13. propget ..... pass
14. propnullns ..... FAIL
(PROPPATCH of property with null namespace (see FAQ))
15. propget ..... FAIL
(No value given for property {}nonamespace)
16. prophighunicode ..... FAIL
(PROPPATCH of property with high unicode value)
17. propget ..... FAIL
(No value given for property
18. propvalnspace ..... FAIL
(PROPPATCH of property with value defining namespace)
19. propwformed ..... pass
20. propinit ..... pass
21. propmanyns ..... pass
22. propget ..... pass
23. propcleanup ..... pass
24. finish ..... pass
<- summary for 'props': of 25 tests run: 20 passed, 5 failed. 80.0%
-> running 'locks':
<- summary for 'locks': of 30 tests run: 0 passed, 30 failed. 0.0%
-> running 'http':
 0. init ..... pass
 1. begin ..... pass
 2. expect100 ..... pass
 3. finish ..... pass
<- summary for 'http': of 4 tests run: 4 passed, 0 failed. 100.0%
See debug.log for network/debug traces.

.
```

Listing A.7: Litmus ohne ACP: Microsoft IIS5.1 Win32

```
-> running 'basic':
 0. init ..... pass
 1. begin ..... pass
 2. options ..... pass
 3. put_get ..... pass
 4. put_get_utf8_segment .. pass
 5. mkcol_over_plain ..... pass
 6. delete ..... pass
 7. delete_null ..... pass
 8. mkcol ..... pass
 9. mkcol_again ..... pass
10. delete_coll ..... pass
11. mkcol_no_parent ..... pass
12. mkcol_with_body ..... pass
13. finish ..... pass
<- summary for 'basic': of 14 tests run: 14 passed, 0 failed. 100.0%
-> running 'copymove':
 0. init ..... pass
 1. begin ..... pass
 2. copy_init ..... pass
 3. copy_simple ..... pass
```

```
 4. copy_overwrite........ pass
 5. copy_cleanup.......... pass
 6. copy_coll............. pass
 7. move.................. pass
 8. move_coll............. pass
 9. move_cleanup.......... pass
10. finish................ pass
<- summary for `copymove': of 11 tests run: 11 passed, 0 failed. 100.0%
-> running `props':
 0. init.................. pass
 1. begin................. pass
 2. propfind_invalid...... pass
 3. propfind_invalid2..... pass
 4. propfind_d0........... pass
 5. propinit.............. pass
 6. propset............... pass
 7. propget............... pass
 8. propmove.............. pass
 9. propget............... pass
10. propdeletes........... pass
11. propget............... pass
12. propreplace........... pass
13. propget............... pass
14. propnullns............ FAIL
(PROPPATCH of property with null namespace (see FAQ))
15. propget............... FAIL
(No value given for property {}nonamespace)
16. prophighunicode....... FAIL
(PROPPATCH of property with high unicode value)
17. propget............... FAIL
18. propvalnspace......... FAIL
(PROPPATCH of property with value defining namespace)
19. propwformed........... pass
20. propinit.............. pass
21. propmanyns............ pass
22. propget............... pass
23. propcleanup........... pass
24. finish................ pass
<- summary for `props': of 25 tests run: 20 passed, 5 failed. 80.0%
-> running `locks':
 0. init.................. pass
 1. begin................. pass
 2. options............... pass
 3. precond............... pass
 4. init_locks............ pass
 5. put................... pass
 6. lock_excl............. pass
 7. discover.............. pass
 8. refresh............... pass
 9. notowner_modify....... WARNING:
MOVE failed with 0 not 423
WARNING: COPY failed with 0 not 423
    ..................... pass (with 2 warnings)
10. notowner_lock......... pass
11. owner_modify.......... pass
12. notowner_modify....... WARNING:
MOVE failed with 412 not 423
WARNING: COPY failed with 0 not 423
    ..................... pass (with 2 warnings)
13. notowner_lock......... pass
14. copy.................. pass
15. cond_put.............. SKIPPED
16. fail_cond_put......... SKIPPED
17. cond_put_with_not..... pass
18. cond_put_corrupt_token FAIL
(conditional PUT with invalid lock-token should fail: 200 OK)
19. complex_cond_put...... SKIPPED
20. fail_complex_cond_put. SKIPPED
21. unlock................ pass
22. fail_cond_put_unlocked pass
23. lock_shared........... pass
24. notowner_modify....... WARNING:
MOVE failed with 412 not 423
    ..................... FAIL
(COPY onto locked resource should fail)
25. notowner_lock......... pass
26. owner_modify.......... pass
27. double_sharedlock..... pass
28. notowner_modify....... WARNING:
MOVE failed with 412 not 423
    ..................... FAIL
(COPY onto locked resource should fail)
29. notowner_lock......... pass
30. unlock................ pass
31. prep_collection....... pass
32. lock_collection....... FAIL
(LOCK on `/Webdav/litmus/lockcoll/': 403 Forbidden)
```

```
33. owner_modify ..... SKIPPED
34. notowner_modify ..... SKIPPED
35. refresh ..... SKIPPED
36. indirect_refresh ..... SKIPPED
37. unlock ..... SKIPPED
38. finish ..... pass
-> 9 tests were skipped.
<- summary for 'locks': of 30 tests run: 26 passed, 4 failed. 86.7%
-> 6 warnings were issued.
-> running 'http':
 0. init ..... pass
 1. begin ..... pass
 2. expect100 ..... pass
 3. finish ..... pass
<- summary for 'http': of 4 tests run: 4 passed, 0 failed. 100.0%

.
```

Listing A.8: Litmus ohne ACP: SharePoint 2003 Server

```
root@ubuntuserver:# litmus http://192.168.79.136/tamino/welcome_4_4_1/ino:dav/ino:dav
-> running 'basic':
 0. init ..... pass
 1. begin ..... pass
 2. options ..... pass
 3. put_get ..... pass
 4. put_get_utf8_segment .. pass
 5. mkcol_over_plain ..... pass
 6. delete ..... pass
 7. delete_null ..... pass

 8. mkcol ..... pass
 9. mkcol_again ..... pass
10. delete_coll ..... pass
11. mkcol_no_parent ..... pass
12. mkcol_with_body ..... FAIL
(MKCOL with weird body must fail with 415 (RFC2518:8.3.1))
13. finish ..... pass
<- summary for 'basic': of 14 tests run: 13 passed, 1 failed. 92.9%
-> running 'copymove':
 0. init ..... pass
 1. begin ..... pass
 2. copy_init ..... pass
 3. copy_simple ..... pass
 4. copy_overwrite ..... pass
 5. copy_cleanup ..... pass
 6. copy_coll ..... pass
 7. move ..... pass
 8. move_coll ..... pass
 9. move_cleanup ..... pass
10. finish ..... pass
<- summary for 'copymove': of 11 tests run: 11 passed, 0 failed. 100.0%
-> running 'props':
 0. init ..... pass
 1. begin ..... pass
 2. propfind_invalid ..... pass
 3. propfind_invalid2 ..... FAIL
 (PROPFIND with invalid namespace declaration in body
 (see FAQ) got 207 response not 400)
 4. propfind_d0 ..... pass
 5. propinit ..... pass
 6. propset ..... pass
 7. propget ..... pass
 8. propmove ..... pass
 9. propget ..... pass
10. propdeletes ..... pass
11. propget ..... pass
12. propreplace ..... pass
13. propget ..... pass
14. propnullns ..... pass
15. propget ..... pass
16. prophighunicode ..... pass
17. propget ..... pass
18. propvalnspace ..... pass
19. propwformed ..... pass
20. propinit ..... pass
21. propmanyns ..... pass
22. propget ..... pass
23. propcleanup ..... pass
24. finish ..... pass
<- summary for 'props': of 25 tests run: 24 passed, 1 failed. 96.0%
-> running 'locks':
 0. init ..... pass
 1. begin ..... pass
```

```
 2. options.............. pass
 3. precond.............. pass
 4. init_locks........... pass
 5. put.................. pass
 6. lock_excl............ pass
 7. discover............. pass
 9. notowner_modify...... pass
 9. notowner_lock........ pass
10. owner_modify......... FAIL (PUT on locked resource failed: 423 Locked)
11. notowner_modify...... pass
12. notowner_lock........ pass
13. copy................. pass
14. cond_put............. pass
15. fail_cond_put........ WARNING: PUT failed with 423 not 412
    ..................... pass (with 1 warning)
16. cond_put_with_not.... pass
17. cond_put_corrupt_token pass
18. complex_cond_put..... pass
19. fail_complex_cond_put. FAIL
(PUT with complex bogus conditional should fail with 412: 423 Locked)
20. unlock............... pass
21. lock_shared.......... pass
22. notowner_modify...... pass
23. notowner_lock........ pass
24. owner_modify......... FAIL
(PUT on locked resource failed: 423 Locked)
25. double_sharedlock.... pass
26. notowner_modify...... FAIL
(DELETE of locked resource should fail)
27. notowner_lock........ FAIL
(LOCK on locked resource)
28. unlock............... FAIL
(UNLOCK on '/tamino/welcome_4_4_1/ino:dav/ino:dav/litmus2/lockme': 423 Locked)
29. finish............... pass
<- summary for 'locks': of 30 tests run: 24 passed, 6 failed. 80.0%
-> 1 warning was issued.
-> running 'http':
 0. init................. pass
 1. begin................ pass
 2. expect100............ pass
 3. finish............... pass
<- summary for 'http': of 4 tests run: 4 passed, 0 failed. 100.0%
root@ubuntuserver:#

.
```

Listing A.9: Litmus ohne ACP: Tamino Win32

A.2 Litmus mit ACP

```
ubuntu@ubuntuserver:~# litmus http://192.168.27.130/webdav webdav webdav
-> running `basic':
 0. init.................. pass
 1. begin................. pass
 2. options............... pass
 3. put_get............... pass
 4. put_get_utf8_segment.. pass
 5. mkcol_over_plain...... pass
 6. delete................ pass
 7. delete_null........... pass
 8. mkcol................. pass
 9. mkcol_again........... pass
10. delete_coll........... pass
11. mkcol_no_parent....... pass
12. mkcol_with_body....... pass
13. finish................ pass
<- summary for `basic': of 14 tests run: 14 passed, 0 failed. 100.0%
-> running `copymove':
 0. init.................. pass
 1. begin................. pass
 2. copy_init............. pass
 3. copy_simple........... pass
 4. copy_overwrite........ pass
 5. copy_cleanup.......... pass
 6. copy_coll............. pass
 7. move.................. pass
 8. move_coll............. pass
 9. move_cleanup.......... pass
10. finish................ pass
<- summary for `copymove': of 11 tests run: 11 passed, 0 failed. 100.0%
-> running `props':
 0. init.................. pass
 1. begin................. pass
 2. propfind_invalid...... pass
 3. propfind_invalid2..... pass
 4. propfind_d0........... pass
 5. propinit.............. pass
 6. propset............... pass
 7. propget............... pass
 8. propmove.............. pass
 9. propget............... pass
10. propdeletes........... pass
11. propget............... pass
12. propreplace........... pass
13. propget............... pass
14. propnullns............ pass
15. propget............... pass
16. prophighunicode....... pass
17. propget............... pass
18. propvalnspace......... pass
19. propwformed........... pass
20. propinit.............. pass
21. propmanyns............ pass
22. propget............... pass
23. propcleanup........... pass
24. finish................ pass
<- summary for `props': of 25 tests run: 25 passed, 0 failed. 100.0%
-> running `locks':
 0. init.................. pass
 1. begin................. pass
 2. options............... pass
 3. precond............... pass
 4. init_locks............ pass
 5. put................... pass
 6. lock_excl............. pass
 7. discover.............. pass
 8. notowner_modify....... pass
 9. notowner_lock......... pass
10. owner_modify.......... pass
11. notowner_modify....... pass
12. notowner_lock......... pass
13. copy.................. pass
14. cond_put.............. pass
15. fail_cond_put......... pass
16. cond_put_with_not..... pass
17. cond_put_corrupt_token WARNING: PUT failed with 400 not 423
    ..................... pass (with 1 warning)
18. complex_cond_put...... pass
19. fail_complex_cond_put. pass
20. unlock................ pass
21. lock_shared........... pass
22. notowner_modify....... pass
23. notowner_lock......... pass
24. owner_modify.......... pass
```

```
25. double_sharedlock..... pass
26. notowner_modify....... pass
27. notowner_lock......... pass
28. unlock................ pass
29. finish................ pass
<- summary 'locks': of 30 tests run: 30 passed, 0 failed. 100.0%
-> 1 warning was issued.
-> running 'http':
 0. init................... pass
 1. begin.................. pass
 2. expect100.............. pass
 3. finish................. pass
<- summary 'http': of 4 tests run: 4 passed, 0 failed. 100.0%
-> running 'acl':
 0. init................... pass
 1. begin.................. pass
 2. acl_init............... pass
 3. acl_get_owner......... FAIL
 (ACL - Retrieving owner element. Error in DAV:owner)
 4. acl_set_owner......... FAIL
 (ACL - Propset on owner element)
 5. acl_get_owner......... FAIL
 (ACL - Retrieving owner element. Error in DAV:owner)
 6. acl_copy_simple....... pass
 7. acl_privilege_set..... FAIL
 (ACL - Retrieving supported-privilege-set element.
 Propfind of DAV:supported-privilege-set)
 8. acl_user_privilege_set FAIL
 (ACL - Retrieving current-user-privilege-set  element.
 Propfind of DAV:current-user-privilege-set )
 9. acl_propfind_set...... FAIL
 (ACL - Retrieving acl element. Finding a resource's ACL)
10. acl_restrictions...... FAIL
(ACL - Retrieving acl-restrictions element.
Error in DAV:acl-restrictions)
11. acl_inherited_acl_set. FAIL
(ACL - Retrieving inherited-acl-set element.
Error in DAV:inherited-acl-set)
12. acl_pcpl_collectn_set. FAIL
(ACL - Retrieving principal-collection-set element.
Error in DAV:principal-collection-set)
13. acl_pcpl_prop_report.. FAIL
(Error in REPORT acl-principal-prop-set)
14. acl_pcpl_match_report. FAIL
(Error in REPORT acl-principal-prop-set)
15. acl_get............... pass
16. acl_set_privileges.... FAIL
(ACL - Trying to set ACEs)
17. acl_privileges_test... FAIL
(ACL - Trying to set ACEs)
18. acl_privileges_f...... FAIL
(ACL - Trying to set ACEs)
19. acl_copy_simple....... WARNING:
Resource COPY that should have failed. We dont have perms
    ..................... pass (with 1 warning)
20. aclcleanup............ pass
21. finish................ pass
<- summary for 'acl': of 22 tests run: 8 passed, 14 failed. 36.4%
-> 1 warning was issued.
See debug.log for network/debug traces.
ubuntu@ubuntuserver:~#
```

Listing A.10: Litmus mit ACP: Apache2 Linux

```
ubuntu@ubuntuserver:~# litmus http://192.168.27.132/webdav webdav webdav
-> running 'basic':
 0. init................... pass
 1. begin.................. pass
 2. options................ pass
 3. put_get................ pass
 4. put_get_utf8_segment.. pass
 5. mkcol_over_plain...... pass
 6. delete................. pass
 7. delete_null............ pass
 8. mkcol.................. pass
 9. mkcol_again............ pass
10. delete_coll............ pass
11. mkcol_no_parent....... pass
12. mkcol_with_body....... pass
13. finish................. pass
<- summary for 'basic': of 14 tests run: 14 passed, 0 failed. 100.0%
-> running 'copymove':
 0. init................... pass
 1. begin.................. pass
```

```
 2. copy_init ..... pass
 3. copy_simple ..... pass
 4. copy_overwrite ..... pass
 5. copy_cleanup ..... pass
 6. copy_coll ..... pass
 7. move ..... pass
 8. move_coll ..... pass
 9. move_cleanup ..... pass
10. finish ..... pass
<- summary for 'copymove': of 11 tests run: 11 passed, 0 failed. 100.0%
-> running 'props':
 0. init ..... pass
 1. begin ..... pass
 2. propfind_invalid ..... pass
 3. propfind_invalid2 ..... pass
 4. propfind_d0 ..... pass
 5. propinit ..... pass
 6. propset ..... pass
 7. propget ..... pass
 8. propmove ..... pass
 9. propget ..... pass
10. propdeletes ..... pass
11. propget ..... pass
12. propreplace ..... pass
13. propget ..... pass
14. propnullns ..... pass
15. propget ..... pass
16. prophighunicode ..... pass
17. propget ..... pass
18. propvalnspace ..... pass
19. propwformed ..... pass
20. propinit ..... pass
21. propmanyns ..... pass
22. propget ..... pass
23. propcleanup ..... pass
24. finish ..... pass
<- summary for 'props': of 25 tests run: 25 passed, 0 failed. 100.0%
-> running 'locks':
 0. init ..... pass
 1. begin ..... pass
 2. options ..... pass
 3. precond ..... pass
 4. init_locks ..... pass
 5. put ..... pass
 6. lock_excl ..... pass
 7. discover ..... pass
 8. notowner_modify ..... pass
 9. notowner_lock ..... pass
10. owner_modify ..... pass
11. notowner_modify ..... pass
12. notowner_lock ..... pass
13. copy ..... pass
14. cond_put ..... pass
15. fail_cond_put ..... pass
16. cond_put_with_not ..... pass
17. cond_put_corrupt_token WARNING: PUT failed with 400 not 423
    ..... pass (with 1 warning)
18. complex_cond_put ..... pass
19. fail_complex_cond_put. pass
20. unlock ..... pass
21. lock_shared ..... pass
22. notowner_modify ..... pass
23. notowner_lock ..... pass
24. owner_modify ..... pass
25. double_sharedlock ..... pass
26. notowner_modify ..... pass
27. notowner_lock ..... pass
28. unlock ..... pass
29. finish ..... pass
<- summary for 'locks': of 30 tests run: 30 passed, 0 failed. 100.0%
-> 1 warning was issued.
-> running 'http':
 0. init ..... pass
 1. begin ..... pass
 2. expect100 ..... pass
 3. finish ..... pass
<- summary for 'http': of 4 tests run: 4 passed, 0 failed. 100.0%
-> running 'acl':
 0. init ..... pass
 1. begin ..... pass
 2. acl_init ..... pass
 3. acl_get_owner ..... FAIL
 (ACL - Retrieving owner element. Error in DAV:owner)
 4. acl_set_owner ..... FAIL
 (ACL - Propset on owner element)
 5. acl_get_owner ..... FAIL
 (ACL - Retrieving owner element. Error in DAV:owner)
```

```
 6. acl_copy_simple....... pass
 7. acl_privilege_set..... FAIL
 (ACL - Retrieving supported-privilege-set element.
 Propfind of DAV:supported-privilege-set)
 8. acl_user_privilege_set FAIL
 (ACL - Retrieving current-user-privilege-set  element.
 Propfind of DAV:current-user-privilege-set )
 9. acl_propfind_set...... FAIL
 (ACL - Retrieving acl element. Finding a resource's ACL)
10. acl_restrictions...... FAIL
(ACL - Retrieving acl-restrictions element.
Error in DAV:acl-restrictions)
11. acl_inherited_acl_set. FAIL
(ACL - Retrieving inherited-acl-set element.
Error in DAV:inherited-acl-set)
12. acl_pcpl_collectn_set. FAIL
(ACL - Retrieving principal-collection-set element.
Error in DAV:principal-collection-set)
13. acl_pcpl_prop_report.. FAIL
(Error in REPORT acl-principal-prop-set)
14. acl_pcpl_match_report. FAIL
(Error in REPORT acl-principal-prop-set)
15. acl_get............... pass
16. acl_set_privileges.... FAIL
(ACL - Trying to set ACEs)
17. acl_privileges_test... FAIL
(ACL - Trying to set ACEs)
18. acl_privileges_f...... FAIL
(ACL - Trying to set ACEs)
19. acl_copy_simple....... WARNING:
Resource COPY that should have failed. We dont have perms
    ..................... pass (with 1 warning)
20. aclcleanup............ pass
21. finish................ pass
<- summary for 'acl': of 22 tests run: 8 passed, 14 failed. 36.4%
See debug.log for network/debug traces.
ubuntu@ubuntuserver:~#
```

Listing A.11: Litmus mit ACP: Apache2 Win32

```
ubuntu@ubuntuserver:~# litmus http://192.168.27.135/dfCatacomb datafinder 123
-> running 'basic':
 0. init................... pass
 1. begin.................. pass
 2. options................ pass
 3. put_get................ pass
 4. put_get_utf8_segment.. pass
 5. mkcol_over_plain...... pass
 6. delete................. pass
 7. delete_null............ pass
 8. mkcol.................. pass
 9. mkcol_again............ pass
10. delete_coll............ pass
11. mkcol_no_parent....... pass
12. mkcol_with_body....... pass
13. finish................. pass
<- summary for 'basic': of 14 tests run: 14 passed, 0 failed. 100.0%
-> running 'copymove':
 0. init................... pass
 1. begin.................. pass
 2. copy_init.............. pass
 3. copy_simple............ pass
 4. copy_overwrite........ pass
 5. copy_cleanup.......... pass
 6. copy_coll.............. pass
 7. move................... pass
 8. move_coll.............. pass
 9. move_cleanup.......... pass
10. finish................. pass
<- summary for 'copymove': of 11 tests run: 11 passed, 0 failed. 100.0%
-> running 'props':
 0. init................... pass
 1. begin.................. pass
 2. propfind_invalid...... pass
 3. propfind_invalid2..... pass
 4. propfind_d0........... pass
 5. propinit.............. pass
 6. propset................ pass
 7. propget................ pass
 8. propmove............... pass
 9. propget................ pass
10. propdeletes........... pass
11. propget................ pass
12. propreplace........... pass
```

```
13. propget.............. pass
14. propnullns........... pass
15. propget.............. pass
16. prophighunicode...... pass
17. propget.............. pass
18. propvalnspace........ pass
19. propwformed.......... pass
20. propinit............. pass
21. propmanyns........... pass
22. propget.............. pass
23. propcleanup.......... pass
24. finish............... pass
<- summary for 'props': of 25 tests run: 25 passed, 0 failed. 100.0%
-> running 'locks':
 0. init................. pass
 1. begin................ pass
 2. options.............. pass
 3. precond.............. pass
 4. init_locks........... pass
 5. put.................. pass
 6. lock_excl............ pass
 7. discover............. pass
 8. notowner_modify...... pass
 9. notowner_lock........ pass
10. owner_modify......... pass
11. notowner_modify...... pass
12. notowner_lock........ pass
13. copy................. pass
14. cond_put............. pass
15. fail_cond_put........ WARNING:
PUT failed with 423 not 412
    ..................... pass (with 1 warning)
16. cond_put_with_not.... pass
17. cond_put_corrupt_token WARNING:
PUT failed with 400 not 423
    ..................... pass (with 1 warning)
18. complex_cond_put..... pass
19. fail_complex_cond_put. pass
20. unlock............... pass
21. lock_shared.......... pass
22. notowner_modify...... pass
23. notowner_lock........ pass
24. owner_modify......... pass
25. double_sharedlock.... pass
26. notowner_modify...... pass
27. notowner_lock........ pass
28. unlock............... pass
29. finish............... pass
<- summary for 'locks': of 30 tests run: 30 passed, 0 failed. 100.0%
-> 2 warnings were issued.
-> running 'http':
 0. init................. pass
 1. begin................ pass
 2. expect100............ pass
 3. finish............... pass
<- summary for 'http': of 4 tests run: 4 passed, 0 failed. 100.0%
-> running 'acl':
 0. init................. pass
 1. begin................ pass
 2. acl_init............. pass
 3. acl_get_owner........ pass
 4. acl_set_owner........ pass
 5. acl_get_owner........ pass
 6. acl_copy_simple...... pass
 7. acl_privilege_set.... pass
 8. acl_user_privilege_set pass
 9. acl_propfind_set..... pass
10. acl_restrictions..... pass
11. acl_inherited_acl_set. pass
12. acl_pcpl_collectn_set. pass
13. acl_pcpl_prop_report.. FAIL (Error in REPORT acl-principal-prop-set)
14. acl_pcpl_match_report. FAIL (Error in REPORT acl-principal-prop-set)
15. acl_get.............. pass
16. acl_set_privileges.... pass
17. acl_privileges_test... pass
18. acl_privileges_f..... pass
19. acl_copy_simple...... WARNING:
Resource COPY that should have failed. We dont have perms
    ..................... pass (with 1 warning)
20. aclcleanup........... pass
21. finish............... pass
<- summary for 'acl': of 22 tests run: 20 passed, 2 failed. 90.9%
-> 1 warning was issued.
ubuntu@ubuntuserver:~#

.
```

Listing A.12: Litmus mit ACP: Catacomb mit ACP Linux

```
ubuntu@ubuntuserver:~# litmus http://192.168.239.144/dav root root
-> running `basic':
 0. init.................. pass
 1. begin................. pass
 2. options............... pass
 3. put_get............... pass
 4. put_get_utf8_segment.. pass
 5. mkcol_over_plain...... pass
 6. delete................ pass
 7. delete_null........... pass
 8. mkcol................. pass
 9. mkcol_again........... pass
10. delete_coll........... pass
11. mkcol_no_parent....... pass
12. mkcol_with_body....... pass
13. finish................ pass
<- summary for `basic': of 14 tests run: 14 passed, 0 failed. 100.0%
-> running `copymove':
 0. init.................. pass
 1. begin................. pass
 2. copy_init............. pass
 3. copy_simple........... pass
 4. copy_overwrite........ WARNING:
 COPY to existing resource didn't give 204
    ....................... pass (with 1 warning)
 5. copy_cleanup.......... pass
 6. copy_coll............. pass
 7. move.................. WARNING:
 MOVE to existing collection resource didn't give 204
    ....................... pass (with 1 warning)
 8. move_coll............. pass
 9. move_cleanup.......... pass
10. finish................ pass
<- summary for `copymove': of 11 tests run: 11 passed, 0 failed. 100.0%
-> 2 warnings were issued.
-> running `props':
 0. init.................. pass
 1. begin................. pass
 2. propfind_invalid...... pass
 3. propfind_invalid2..... pass
 4. propfind_d0........... pass
 5. propinit.............. pass
 6. propset............... pass
 7. propget............... pass
 8. propmove.............. pass
 9. propget............... pass
10. propdeletes........... pass
11. propget............... pass
12. propreplace........... pass
13. propget............... pass
14. propnullns............ pass
15. propget............... pass
16. prophighunicode....... pass
17. propget............... pass
18. propvalnspace......... pass
19. propwformed........... pass
20. propinit.............. pass
21. propmanyns............ pass
22. propget............... pass
23. propcleanup........... pass
24. finish................ pass
<- summary for `props': of 25 tests run: 25 passed, 0 failed. 100.0%
-> running `locks':
 0. init.................. pass
 1. begin................. pass
 2. options............... pass
 3. precond............... pass
 4. init_locks............ pass
 5. put................... pass
 6. lock_excl............. pass
 7. discover.............. pass
 8. notowner_modify....... pass
 9. notowner_lock......... pass
10. owner_modify.......... pass
11. notowner_modify....... pass
12. notowner_lock......... pass
13. copy.................. pass
14. cond_put.............. pass
15. fail_cond_put......... FAIL
(conditional PUT with invalid lock-token should fail: 200 OK)
16. cond_put_with_not..... pass
```

```
17. cond_put_corrupt_token WARNING: PUT failed with 200 not 423
    ..................... pass (with 1 warning)
18. complex_cond_put..... pass
19. fail_complex_cond_put. FAIL
(PUT with complex bogus conditional should fail with 412: 200 OK)
20. unlock.............. pass
21. lock_shared......... pass
22. notowner_modify..... pass
23. notowner_lock....... pass
24. owner_modify........ pass
25. double_sharedlock... FAIL (shared LOCK on locked resource:
423 Locked)
26. notowner_modify..... pass
27. notowner_lock....... pass
28. unlock.............. pass
29. finish.............. pass
<- summary for 'locks': of 30 tests run: 27 passed, 3 failed. 90.0%
-> running 'http':
 0. init................ pass
 1. begin............... pass
 2. expect100........... pass
 3. finish.............. pass
<- summary for 'http': of 4 tests run: 4 passed, 0 failed. 100.0%
-> running 'acl':
 0. init................ pass
 1. begin............... pass
 2. acl_init............ pass
 3. acl_get_owner....... FAIL
 (ACL - Retrieving owner element. Error in DAV:owner)
 4. acl_set_owner....... FAIL
 (ACL - Propset on owner element)
 5. acl_get_owner....... FAIL
 (ACL - Retrieving owner element. Error in DAV:owner)
 6. acl_copy_simple..... pass
 7. acl_privilege_set... FAIL
 (ACL - Retrieving supported-privilege-set element.
 Propfind of DAV:supported-privilege-set)
 8. acl_user_privilege_set FAIL
 (ACL - Retrieving current-user-privilege-set  element.
 Propfind of DAV:current-user-privilege-set )
 9. acl_propfind_set.... FAIL
 (ACL - Retrieving acl element. Finding a resource's ACL)
10. acl_restrictions.... FAIL
(ACL - Retrieving acl-restrictions element.
Error in DAV:acl-restrictions)
11. acl_inherited_acl_set. FAIL
(ACL - Retrieving inherited-acl-set element.
Error in DAV:inherited-acl-set)
12. acl_pcpl_collectn_set. FAIL
(ACL - Retrieving principal-collection-set element.
Error in DAV:principal-collection-set)
13. acl_pcpl_prop_report.. FAIL
(Error in REPORT acl-principal-prop-set)
14. acl_pcpl_match_report. FAIL
(Error in REPORT acl-principal-prop-set)
15. acl_get............. pass
16. acl_set_privileges.... FAIL
(ACL - Trying to set ACEs)
17. acl_privileges_test... FAIL
(ACL - Trying to set ACEs)
18. acl_privileges_f..... FAIL
(ACL - Trying to set ACEs)
19. acl_copy_simple..... WARNING:
Resource COPY that should have failed. We dont have perms
    ..................... pass (with 1 warning)
20. aclcleanup.......... pass
21. finish.............. pass
<- summary for 'acl': of 22 tests run: 8 passed, 14 failed. 36.4%
-> 1 warning was issued.
See debug.log for network/debug traces.
ubuntu@ubuntuserver:~#

.
```

Listing A.13: Litmus mit ACP: Lighttpd Linux

```
ubuntu@ubuntuserver:# litmus http://192.168.27.139:8080/slide/files root root
-> running 'basic':
 0. init................ pass
 1. begin............... pass
 2. options............. pass
 3. put_get............. pass
 4. put_get_utf8_segment.. pass
 5. mkcol_over_plain..... pass
```

```
 6. delete ..... pass
 7. delete_null ..... pass
 8. mkcol ..... pass
 9. mkcol_again ..... pass
10. delete_coll ..... pass
11. mkcol_no_parent ..... pass
12. mkcol_with_body ..... pass
13. finish ..... pass
<- summary for 'basic': of 14 tests run: 14 passed, 0 failed. 100.0%
-> running 'copymove':
 0. init ..... pass
 1. begin ..... pass
 2. copy_init ..... pass
 3. copy_simple ..... pass
 4. copy_overwrite ..... pass
 5. copy_cleanup ..... pass
 6. copy_coll ..... pass
 7. move ..... pass
 8. move_coll ..... pass
 9. move_cleanup ..... pass
10. finish ..... pass
<- summary for 'copymove': of 11 tests run: 11 passed, 0 failed. 100.0%
-> running 'props':
 0. init ..... pass
 1. begin ..... pass
 2. propfind_invalid ..... pass
 3. propfind_invalid2 ..... pass
 4. propfind_d0 ..... pass
 5. propinit ..... pass
 6. propset ..... pass
 7. propget ..... pass
 8. propmove ..... pass
 9. propget ..... pass
10. propdeletes ..... pass
11. propget ..... pass
12. propreplace ..... pass
13. propget ..... pass
14. propnullns ..... pass
15. propget ..... pass
16. prophighunicode ..... pass
17. propget ..... pass
18. propvalnspace ..... pass
19. propwformed ..... pass
20. propinit ..... pass
21. propmanyns ..... pass
22. propget ..... pass
23. propcleanup ..... pass
24. finish ..... pass
<- summary for 'props': of 25 tests run: 25 passed, 0 failed. 100.0%
-> running 'locks':
 0. init ..... pass
 1. begin ..... pass
 2. options ..... pass
 3. precond ..... pass
 4. init_locks ..... pass
 5. put ..... pass
 6. lock_excl ..... pass
 7. discover ..... pass
 8. notowner_modify ..... pass
 9. notowner_lock ..... pass
10. owner_modify ..... pass
11. notowner_modify ..... pass
12. notowner_lock ..... pass
13. copy ..... pass
14. cond_put ..... pass
15. fail_cond_put ..... WARNING: PUT failed with 423 not 412
    ..... pass (with 1 warning)
16. cond_put_with_not ..... pass
17. cond_put_corrupt_token pass
18. complex_cond_put ..... pass
19. fail_complex_cond_put. FAIL
(PUT with complex bogus conditional should fail with 412: 204 No Content)
20. unlock ..... pass
21. lock_shared ..... pass
22. notowner_modify ..... pass
23. notowner_lock ..... pass
24. owner_modify ..... pass
25. double_sharedlock ..... pass
26. notowner_modify ..... pass
27. notowner_lock ..... pass
28. unlock ..... pass
29. finish ..... pass
<- summary for 'locks': of 30 tests run: 29 passed, 1 failed. 96.7%
-> running 'acl':
 0. init ..... pass
 1. begin ..... pass
 2. acl_init ..... pass
```

```
 3. acl_get_owner......... pass
 4. acl_set_owner......... pass
 5. acl_get_owner......... pass
 6. acl_copy_simple....... pass
 7. acl_privilege_set..... pass
 8. acl_user_privilege_set pass
 9. acl_propfind_set...... pass
10. acl_restrictions...... pass
11. acl_inherited_acl_set. pass
12. acl_pcpl_collectn_set. pass
13. acl_pcpl_prop_report.. pass
14. acl_pcpl_match_report. FAIL (Error in REPORT acl-principal-prop-set)
15. acl_get............... pass
16. acl_set_privileges.... pass
17. acl_privileges_test... pass
18. acl_privileges_f...... pass
19. acl_copy_simple....... WARNING:
Resource COPY that should have failed. We dont have perms
    ....................... pass (with 1 warning)
20. aclcleanup............ pass
21. finish................ pass
<- summary for 'acl': of 22 tests run: 21 passed, 1 failed. 95.5%
-> 1 warning was issued.
See debug.log for network/debug traces.
ubuntu@ubuntuserver:#
```

Listing A.14: Litmus mit ACP: Jakarta Slide2.1 Linux

```
ubuntu@ubuntuserver:# litmus http://192.168.27.138:8080/slide/files root root
-> running 'basic':
 0. init.................. pass
 1. begin................. pass
 2. options............... pass
 3. put_get............... pass
 4. put_get_utf8_segment.. pass
 5. mkcol_over_plain...... pass
 6. delete................ pass
 7. delete_null........... pass
 8. mkcol................. pass
 9. mkcol_again........... pass
10. delete_coll........... pass
11. mkcol_no_parent....... pass
12. mkcol_with_body....... pass
13. finish................ pass
<- summary for 'basic': of 14 tests run: 14 passed, 0 failed. 100.0%
-> running 'copymove':
 0. init.................. pass
 1. begin................. pass
 2. copy_init............. pass
 3. copy_simple........... pass
 4. copy_overwrite........ pass
 5. copy_cleanup.......... pass
 6. copy_coll............. pass
 7. move.................. pass
 8. move_coll............. pass
 9. move_cleanup.......... pass
10. finish................ pass
<- summary for 'copymove': of 11 tests run: 11 passed, 0 failed. 100.0%
-> running 'props':
 0. init.................. pass
 1. begin................. pass
 2. propfind_invalid...... pass
 3. propfind_invalid2..... pass
 4. propfind_d0........... pass
 5. propinit.............. pass
 6. propset............... pass
 7. propget............... pass
 8. propmove.............. pass
 9. propget............... pass
10. propdeletes........... pass
11. propget............... pass
12. propreplace........... pass
13. propget............... pass
14. propnullns............ pass
15. propget............... pass
16. prophighunicode....... pass
17. propget............... pass
18. propvalnspace......... pass
19. propwformed........... pass
20. propinit.............. pass
21. propmanyns............ pass
22. propget............... pass
23. propcleanup........... pass
24. finish................ pass
```

```
<- summary for 'props': of 25 tests run: 25 passed, 0 failed. 100.0%
-> running 'locks':
 0. init.................. pass
 1. begin................. pass
 2. options............... pass
 3. precond............... pass
 4. init_locks............ pass
 5. put................... pass
 6. lock_excl............. pass
 7. discover.............. pass
 8. notowner_modify....... pass
 9. notowner_lock......... pass
10. owner_modify.......... pass
11. notowner_modify....... pass
12. notowner_lock......... pass
13. copy.................. pass
14. cond_put.............. pass
15. fail_cond_put......... WARNING: PUT failed with 423 not 412
    ....................... pass (with 1 warning)
16. cond_put_with_not..... pass
17. cond_put_corrupt_token pass
18. complex_cond_put...... pass
19. fail_complex_cond_put. FAIL
(PUT with complex bogus conditional should fail with 412: 204 No Content)
20. unlock................ pass
21. lock_shared........... pass
22. notowner_modify....... pass
23. notowner_lock......... pass
24. owner_modify.......... pass
25. double_sharedlock..... pass
26. notowner_modify....... pass
27. notowner_lock......... pass
28. unlock................ pass
29. finish................ pass
<- summary for 'locks': of 30 tests run: 29 passed, 1 failed. 96.7%
-> running 'acl':
 0. init.................. pass
 1. begin................. pass
 2. acl_init.............. pass
 3. acl_get_owner......... pass
 4. acl_set_owner......... pass
 5. acl_get_owner......... pass
 6. acl_copy_simple....... pass
 7. acl_privilege_set..... pass
 8. acl_user_privilege_set pass
 9. acl_propfind_set...... pass
10. acl_restrictions...... pass
11. acl_inherited_acl_set. pass
12. acl_pcpl_collectn_set. pass
13. acl_pcpl_prop_report.. pass
14. acl_pcpl_match_report. FAIL (Error in REPORT acl-principal-prop-set)
15. acl_get............... pass
16. acl_set_privileges.... pass
17. acl_privileges_test... pass
18. acl_privileges_f...... pass
19. acl_copy_simple....... WARNING:
Resource COPY that should have failed. We dont have perms
    ....................... pass (with 1 warning)
20. aclcleanup............ pass
21. finish................ pass
<- summary for 'acl': of 22 tests run: 21 passed, 1 failed. 95.5%
-> 1 warning was issued.
See debug.log for network/debug traces.
ubuntu@ubuntuserver:#
```

Listing A.15: Litmus mit ACP: Jakarta Slide2.1 Win32

```
ubuntu@ubuntuserver:~# litmus http://192.168.27.137/webdav webdavuser webdav
-> running 'basic':
 0. init.................. pass
 1. begin................. pass
 2. options............... pass
 3. put_get............... pass
 4. put_get_utf8_segment.. pass
 5. mkcol_over_plain...... pass
 6. delete................ pass
 7. delete_null........... pass
 8. mkcol................. pass
 9. mkcol_again........... pass
10. delete_coll........... pass
11. mkcol_no_parent....... pass
12. mkcol_with_body....... pass
13. finish................ pass
<- summary for 'basic': of 14 tests run: 14 passed, 0 failed. 100.0%
```

```
-> running `copymove':
 0. init.................. pass
 1. begin................. pass
 2. copy_init............. pass
 3. copy_simple........... pass
 4. copy_overwrite........ pass
 5. copy_cleanup.......... pass
 6. copy_coll............. pass
 7. move.................. pass
 8. move_coll............. pass
 9. move_cleanup.......... pass
10. finish................ pass
<- summary for `copymove': of 11 tests run: 11 passed, 0 failed. 100.0%
-> running `props':
 0. init.................. pass
 1. begin................. pass
 2. propfind_invalid...... pass
 3. propfind_invalid2..... pass
 4. propfind_d0........... pass
 5. propinit.............. pass
 6. propset............... pass
 7. propget............... pass
 8. propmove.............. pass
 9. propget............... pass
10. propdeletes........... pass
11. propget............... pass
12. propreplace........... pass
13. propget............... pass
14. propnullns............ FAIL
(PROPPATCH of property with null namespace (see FAQ))
15. propget............... FAIL
(No value given for property {}nonamespace)
16. prophighunicode....... FAIL
(PROPPATCH of property with high unicode value)
17. propget............... FAIL
(No value given for property {http://webdav.org/neon/litmus/}high-unicode)
18. propvalnspace......... FAIL
(PROPPATCH of property with value defining namespace)
19. propwformed........... pass
20. propinit.............. pass
21. propmanyns............ pass
22. propget............... pass
23. propcleanup........... pass
24. finish................ pass
<- summary for `props': of 25 tests run: 20 passed, 5 failed. 80.0%
-> running `acl':
 0. init.................. pass
 1. begin................. pass
 2. acl_init.............. pass
 3. acl_get_owner......... FAIL
 (ACL - Retrieving owner element. Error in DAV:owner)
 4. acl_set_owner......... FAIL
 (ACL - Propset on owner element)
 5. acl_get_owner......... FAIL
 (ACL - Retrieving owner element. Error in DAV:owner)
 6. acl_copy_simple....... pass
 7. acl_privilege_set..... FAIL
 (ACL - Retrieving supported-privilege-set element.
 Propfind of DAV:supported-privilege-set)
 8. acl_user_privilege_set FAIL
 (ACL - Retrieving current-user-privilege-set element.
 Propfind of DAV:current-user-privilege-set )
 9. acl_propfind_set...... FAIL
 (ACL - Retrieving acl element. Finding a resource's ACL)
10. acl_restrictions...... FAIL
(ACL - Retrieving acl-restrictions element. Error in DAV:acl-restrictions)
11. acl_inherited_acl_set. FAIL
(ACL - Retrieving inherited-acl-set element. Error in DAV:inherited-acl-set)
12. acl_pcpl_collectn_set. FAIL
(ACL - Retrieving principal-collection-set element.
Error in DAV:principal-collection-set)
13. acl_pcpl_prop_report.. FAIL
(Error in REPORT acl-principal-prop-set)
14. acl_pcpl_match_report. FAIL
(Error in REPORT acl-principal-prop-set)
15. acl_get............... pass
16. acl_set_privileges.... FAIL
(ACL - Trying to set ACEs)
17. acl_privileges_test... FAIL
(ACL - Trying to set ACEs)
18. acl_privileges_f...... FAIL
(ACL - Trying to set ACEs)
19. acl_copy_simple....... WARNING:
Resource COPY that should have failed. We dont have perms
    ..................... pass (with 1 warning)
20. aclcleanup............ pass
21. finish................ pass
```

```
<- summary for 'acl': of 22 tests run: 8 passed, 14 failed. 36.4%
-> 1 warning was issued.
See debug.log for network/debug traces.

.
```

Listing A.16: Litmus mit ACP: Microsoft IIS5.1 Win32

```
-> running 'basic':
 0. init.................. pass
 1. begin................. pass
 2. options............... pass
 3. put_get............... pass
 4. put_get_utf8_segment.. pass
 5. mkcol_over_plain...... pass
 6. delete................ pass
 7. delete_null........... pass
 8. mkcol................. pass
 9. mkcol_again........... pass
10. delete_coll........... pass
11. mkcol_no_parent....... pass
12. mkcol_with_body....... pass
13. finish................ pass
<- summary for 'basic': of 14 tests run: 14 passed, 0 failed. 100.0%
-> running 'copymove':
 0. init.................. pass
 1. begin................. pass
 2. copy_init............. pass
 3. copy_simple........... pass
 4. copy_overwrite........ pass
 5. copy_cleanup.......... pass
 6. copy_coll............. pass
 7. move.................. pass
 8. move_coll............. pass
 9. move_cleanup.......... pass
10. finish................ pass
<- summary for 'copymove': of 11 tests run: 11 passed, 0 failed. 100.0%
-> running 'props':
 0. init.................. pass
 1. begin................. pass
 2. propfind_invalid...... pass
 3. propfind_invalid2..... pass
 4. propfind_d0........... pass
 5. propinit.............. pass
 6. propset............... pass
 7. propget............... pass
 8. propmove.............. pass
 9. propget............... pass
10. propdeletes........... pass
11. propget............... pass
12. propreplace........... pass
13. propget............... pass
14. propnullns............ FAIL
(PROPPATCH of property with null namespace (see FAQ))
15. propget............... FAIL
(No value given for property {}nonamespace)
16. prophighunicode....... FAIL
(PROPPATCH of property with high unicode value)
17. propget............... FAIL
18. propvalnspace......... FAIL
(PROPPATCH of property with value defining namespace)
19. propwformed........... pass
20. propinit.............. pass
21. propmanyns............ pass
22. propget............... pass
23. propcleanup........... pass
24. finish................ pass
<- summary for 'props': of 25 tests run: 20 passed, 5 failed. 80.0%
-> running 'locks':
 0. init.................. pass
 1. begin................. pass
 2. options............... pass
 3. precond............... pass
 4. init_locks............ pass
 5. put................... pass
 6. lock_excl............. pass
 7. discover.............. pass
 8. refresh............... pass
 9. notowner_modify....... WARNING:
MOVE failed with 0 not 423
WARNING: COPY failed with 0 not 423
    ..................... pass (with 2 warnings)
10. notowner_lock......... pass
11. owner_modify.......... pass
```

```
12. notowner_modify....... WARNING:
MOVE failed with 412 not 423
WARNING: COPY failed with 0 not 423
    ........................ pass (with 2 warnings)
13. notowner_lock......... pass
14. copy.................. pass
15. cond_put.............. SKIPPED
16. fail_cond_put......... SKIPPED
17. cond_put_with_not..... pass
18. cond_put_corrupt_token FAIL
(conditional PUT with invalid lock-token should fail: 200 OK)
19. complex_cond_put...... SKIPPED
20. fail_complex_cond_put. SKIPPED
21. unlock................ pass
22. fail_cond_put_unlocked pass
23. lock_shared........... pass
24. notowner_modify....... WARNING:
MOVE failed with 412 not 423
    ........................ FAIL
(COPY onto locked resource should fail)
25. notowner_lock......... pass
26. owner_modify.......... pass
27. double_sharedlock..... pass
28. notowner_modify....... WARNING:
MOVE failed with 412 not 423
    ........................ FAIL
(COPY onto locked resource should fail)
29. notowner_lock......... pass
30. unlock................ pass
31. prep_collection....... pass
32. lock_collection....... FAIL
(LOCK on '/Webdav/litmus/lockcoll/': 403 Forbidden)
33. owner_modify.......... SKIPPED
34. notowner_modify....... SKIPPED
35. refresh............... SKIPPED
36. indirect_refresh...... SKIPPED
37. unlock................ SKIPPED
38. finish................ pass
-> 9 tests were skipped.
<- summary for 'locks': of 30 tests run: 26 passed, 4 failed. 86.7%
-> 6 warnings were issued.
-> running 'http':
 0. init.................. pass
 1. begin................. pass
 2. expect100............. pass
 3. finish................ pass
<- summary for 'http': of 4 tests run: 4 passed, 0 failed. 100.0%
-> running 'acl':
 0. init.................. pass
 1. begin................. pass
 2. acl_init.............. pass
 3. acl_get_owner......... pass
 4. acl_set_owner......... FAIL (ACL - Propset on owner element)
 5. acl_get_owner......... pass
 6. acl_copy_simple....... pass
 7. acl_privilege_set..... pass
 8. acl_user_privilege_set pass
 9. acl_propfind_set...... pass
10. acl_restrictions...... pass
11. acl_inherited_acl_set. pass
12. acl_pcpl_collectn_set. pass
13. acl_pcpl_prop_report.. FAIL (Error in REPORT acl-principal-prop-set)
14. acl_pcpl_match_report. FAIL (Error in REPORT acl-principal-prop-set)
15. acl_get............... pass
16. acl_set_privileges.... pass
17. acl_privileges_test... FAIL (ACL - Trying to set ACEs)
18. acl_privileges_f...... FAIL (ACL - Trying to set ACEs)
19. acl_copy_simple....... WARNING:
Resource COPY that should have failed. We dont have perms
    ........................ pass (with 1 warning)
20. aclcleanup............ pass
21. finish................ pass
<- summary for 'acl': of 22 tests run: 17 passed, 5 failed. 77.3%
-> 1 warning was issued.

.
```

Listing A.17: Litmus mit ACP: SharePoint 2003 Server

```
root@ubuntuserver:# litmus http://192.168.79.136/tamino/welcome_4_4_1/ino:dav/ino:dav
-> running 'basic':
 0. init.................. pass
 1. begin................. pass
 2. options............... pass
```

```
 3. put_get ..... pass
 4. put_get_utf8_segment .. pass
 5. mkcol_over_plain ..... pass
 6. delete ..... pass
 7. delete_null ..... pass

 8. mkcol ..... pass
9. mkcol_again ..... pass
10. delete_coll ..... pass
11. mkcol_no_parent ..... pass
12. mkcol_with_body ..... FAIL (MKCOL with weird body must fail with 415 (RFC2518:8.3.1))
13. finish ..... pass
<- summary for `basic': of 14 tests run: 13 passed, 1 failed. 92.9%
-> running `copymove':
 0. init ..... pass
 1. begin ..... pass
 2. copy_init ..... pass
 3. copy_simple ..... pass
 4. copy_overwrite ..... pass
 5. copy_cleanup ..... pass
 6. copy_coll ..... pass
 7. move ..... pass
 8. move_coll ..... pass
 9. move_cleanup ..... pass
10. finish ..... pass
<- summary for `copymove': of 11 tests run: 11 passed, 0 failed. 100.0%
-> running `props':
 0. init ..... pass
 1. begin ..... pass
 2. propfind_invalid ..... pass
 3. propfind_invalid2 ..... FAIL
 (PROPFIND with invalid namespace declaration in body
 (see FAQ) got 207 response not 400)
 4. propfind_d0 ..... pass
 5. propinit ..... pass
 6. propset ..... pass
 7. propget ..... pass
 8. propmove ..... pass
 9. propget ..... pass
10. propdeletes ..... pass
11. propget ..... pass
12. propreplace ..... pass
13. propget ..... pass
14. propnullns ..... pass
15. propget ..... pass
16. prophighunicode ..... pass
17. propget ..... pass
18. propvalnspace ..... pass
19. propwformed ..... pass
20. propinit ..... pass
21. propmanyns ..... pass
22. propget ..... pass
23. propcleanup ..... pass
24. finish ..... pass
<- summary for `props': of 25 tests run: 24 passed, 1 failed. 96.0%
-> running `locks':
 0. init ..... pass
 1. begin ..... pass
 2. options ..... pass
 3. precond ..... pass
 4. init_locks ..... pass
 5. put ..... pass
 6. lock_excl ..... pass
 7. discover ..... pass
 9. notowner_modify ..... pass
 9. notowner_lock ..... pass
10. owner_modify ..... FAIL (PUT on locked resource failed: 423 Locked)
11. notowner_modify ..... pass
12. notowner_lock ..... pass
13. copy ..... pass
14. cond_put ..... pass
15. fail_cond_put ..... WARNING: PUT failed with 423 not 412
    ..... pass (with 1 warning)
16. cond_put_with_not ..... pass
17. cond_put_corrupt_token pass
18. complex_cond_put ..... pass
19. fail_complex_cond_put. FAIL
(PUT with complex bogus conditional should fail with 412: 423 Locked)
20. unlock ..... pass
21. lock_shared ..... pass
22. notowner_modify ..... pass
23. notowner_lock ..... pass
24. owner_modify ..... FAIL
(PUT on locked resource failed: 423 Locked)
25. double_sharedlock ..... pass
26. notowner_modify ..... FAIL
(DELETE of locked resource should fail)
```

```
27. notowner_lock......... FAIL
(LOCK on locked resource)
28. unlock................ FAIL
(UNLOCK on '/tamino/welcome_4_4_1/ino:dav/ino:dav/litmus2/lockme': 423 Locked)
29. finish................ pass
<- summary for 'locks': of 30 tests run: 24 passed, 6 failed. 80.0%
-> 1 warning was issued.
-> running 'http':
 0. init................... pass
 1. begin.................. pass
 2. expect100.............. pass
 3. finish................. pass
<- summary for 'http': of 4 tests run: 4 passed, 0 failed. 100.0%
-> running 'acl':
 0. init................... pass
 1. begin.................. pass
 2. acl_init............... pass
 3. acl_owner.............. pass
 4. acl_set_owner.......... pass
 5. acl_owner.............. pass
 6. acl_copy_simple........ WARNING:
 Resource COPY that should have failed. We dont have perms
     ...................... pass (with 1 warning)
 7. acl_privilege_set..... pass
 8. acl_user_privilege_set pass
 9. acl_set................ pass
10. acl_restrictions...... pass
11. acl_inherited_acl_set. pass
12. acl_pcpl_collectn_set. pass
13. acl_pcpl_prop_report.. FAIL (Error in REPORT acl-principal-prop-set)
14. acl_pcpl_match_report. FAIL (Error in REPORT acl-principal-prop-set)
15. acl_get............... pass
16. acl_set_privileges.... pass
17. acl_privileges_test... FAIL (ACL - Trying to set ACEs)
18. acl_privileges_f...... pass
19. acl_copy_simple....... WARNING:
Resource COPY that should have failed. We dont have perms
     ...................... pass (with 1 warning)
20. aclcleanup............ pass
21. finish................ pass
<- summary for 'acl': of 22 tests run: 19 passed, 3 failed. 86.4%
-> 2 warnings were issued.
root@ubuntuserver:#

.
```

Listing A.18: Litmus ohne ACP: Tamino Win32

Anhang B

Prestan Testergebnisse

B.1 Prestan ohne ACP

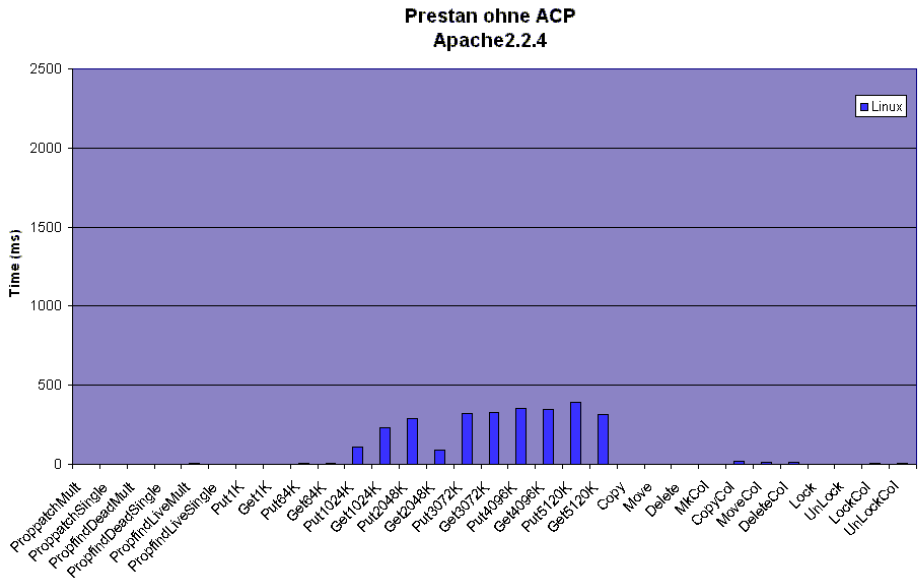

Abbildung B.1: Prestan ohne ACP: Apache2.2.4 Linux

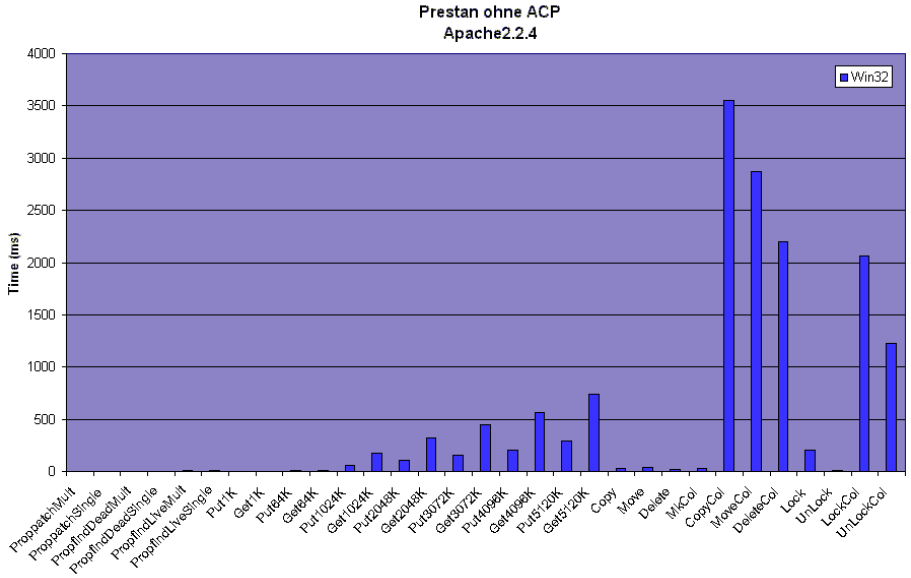

Abbildung B.2: Prestan ohne ACP: Apache2.2.4 Win32

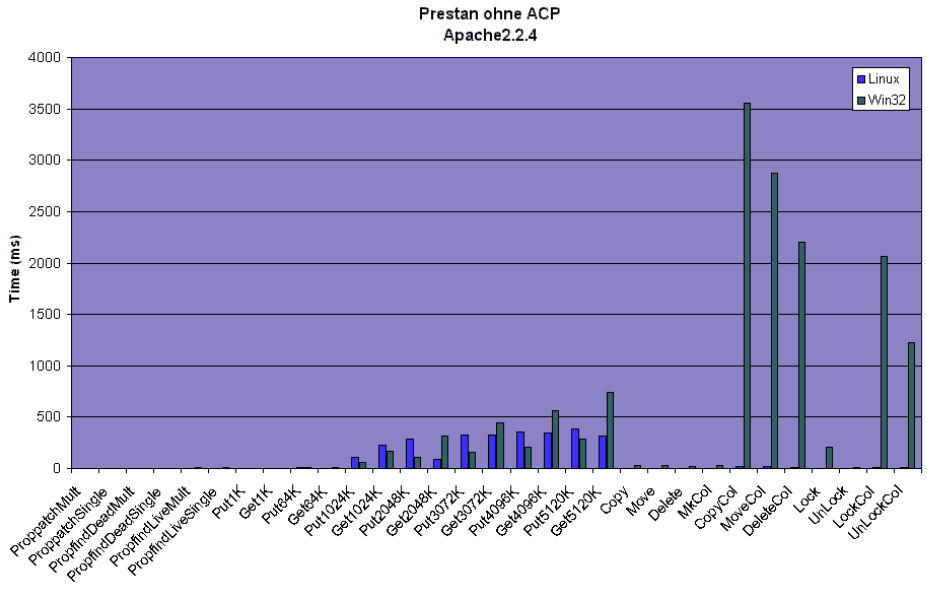

Abbildung B.3: Prestan ohne ACP: Apache2.2.4 Linux/Win32

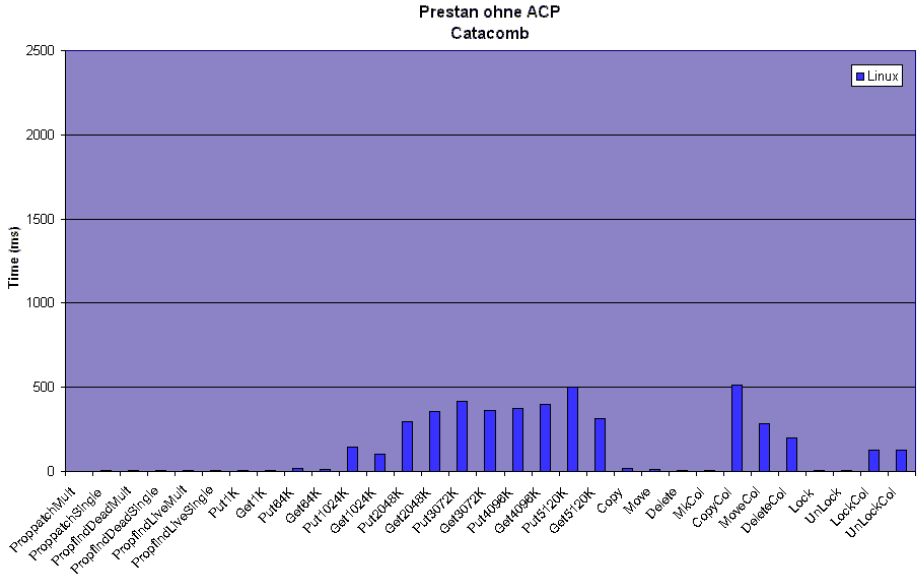

Abbildung B.4: Prestan ohne ACP: Catacomb Linux

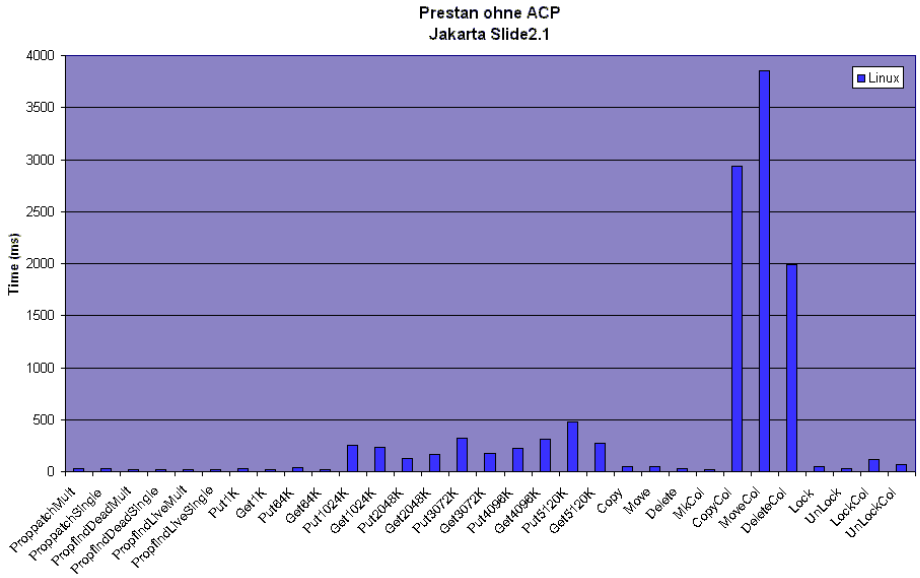

Abbildung B.5: Prestan ohne ACP: Jakarta Slide2.1 Linux

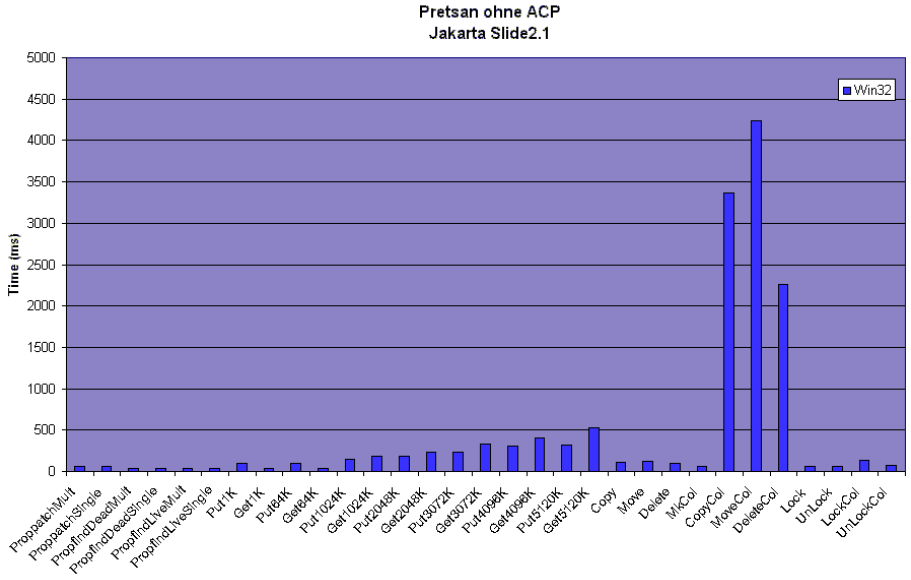

Abbildung B.6: Prestan ohne ACP: Jakarta Slide2.1 Win32

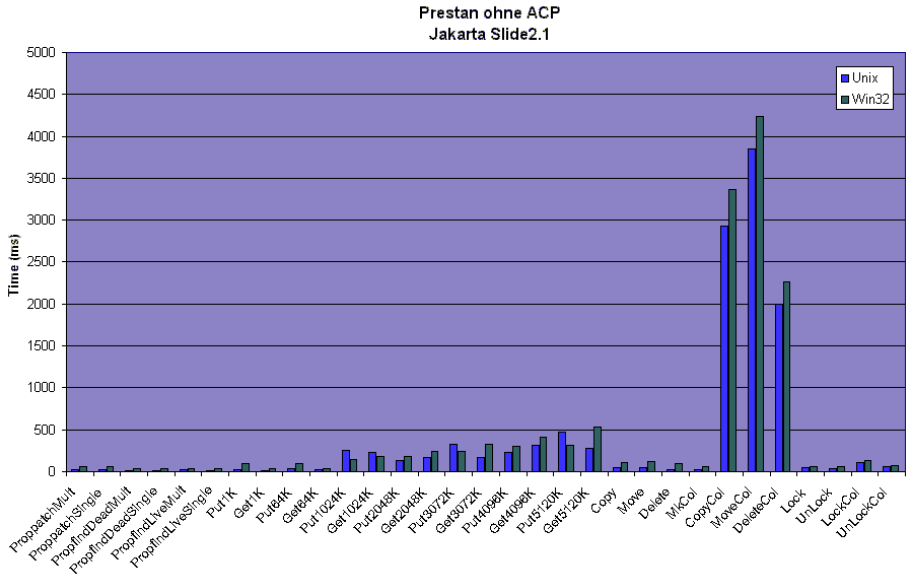

Abbildung B.7: Prestan ohne ACP: Jakarta Slide2.1 Linux/Win32

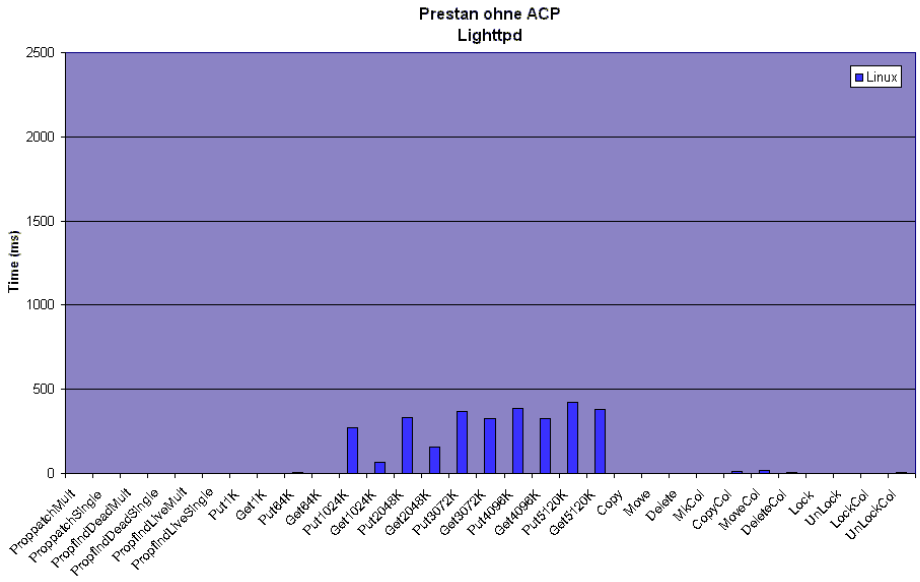

Abbildung B.8: Prestan ohne ACP: Lighttpd1.4 Linux

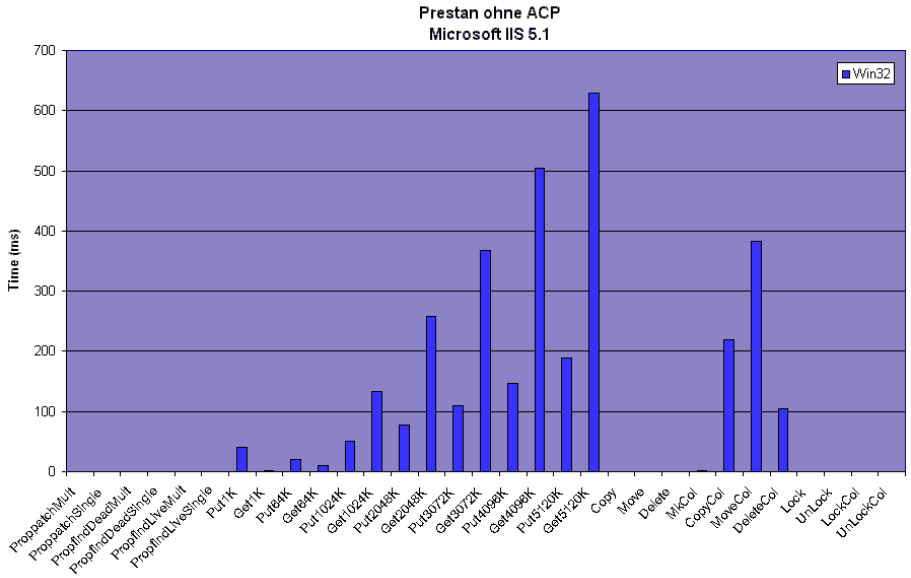

Abbildung B.9: Prestan ohne ACP: Microsoft IIS5.1 Win32

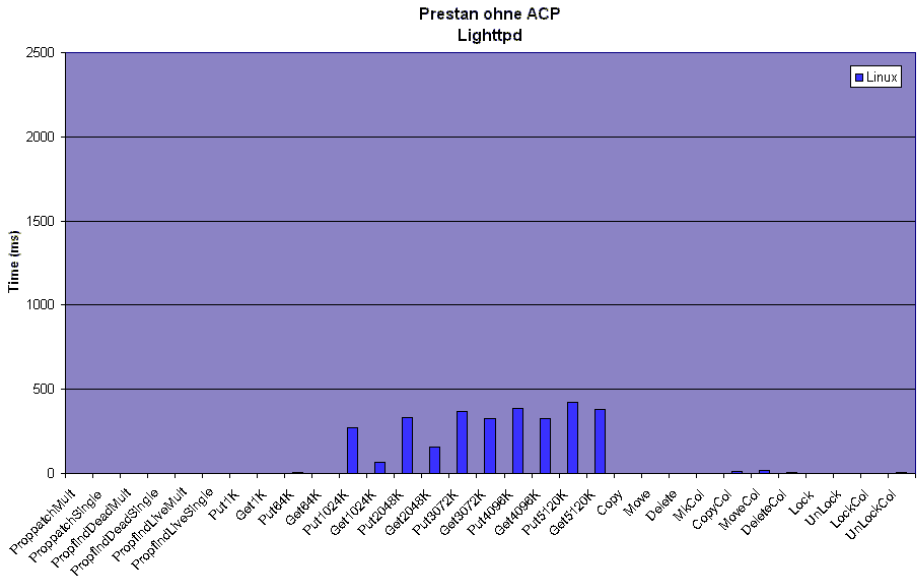

Abbildung B.10: Prestan ohne ACP: SharePoint 2003 Server

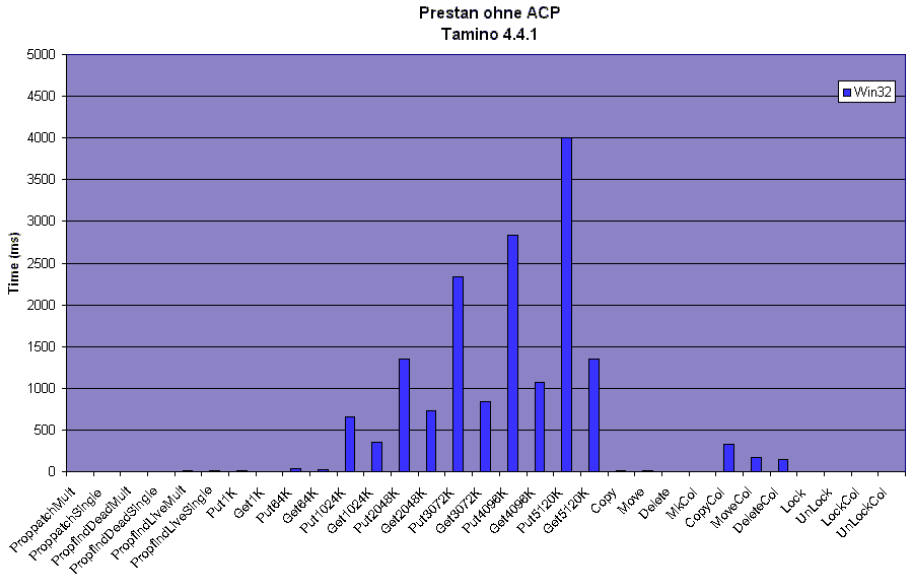

Abbildung B.11: Prestan ohne ACP: Tamino4.4.1 Win32

B.2 Prestan mit ACP

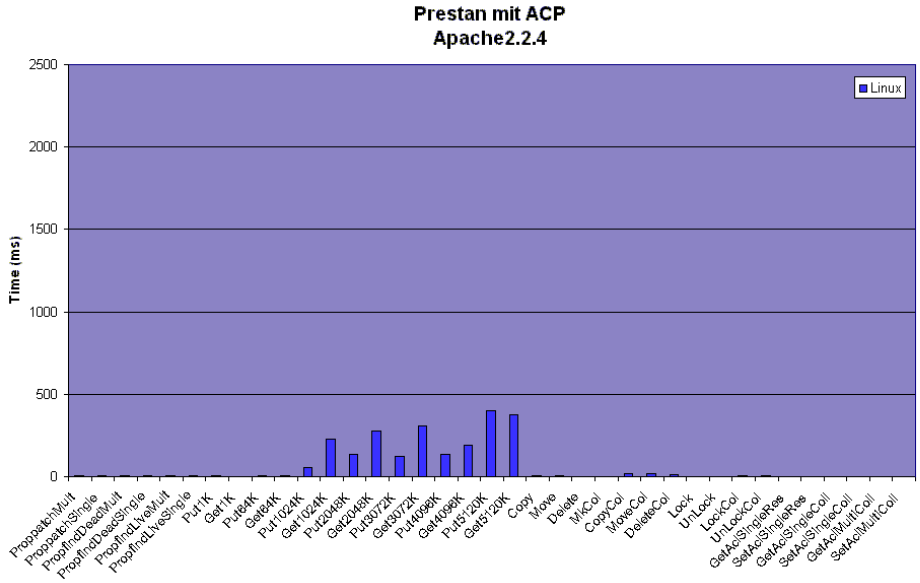

Abbildung B.12: Prestan mit ACP: Apache2.2.4 Linux

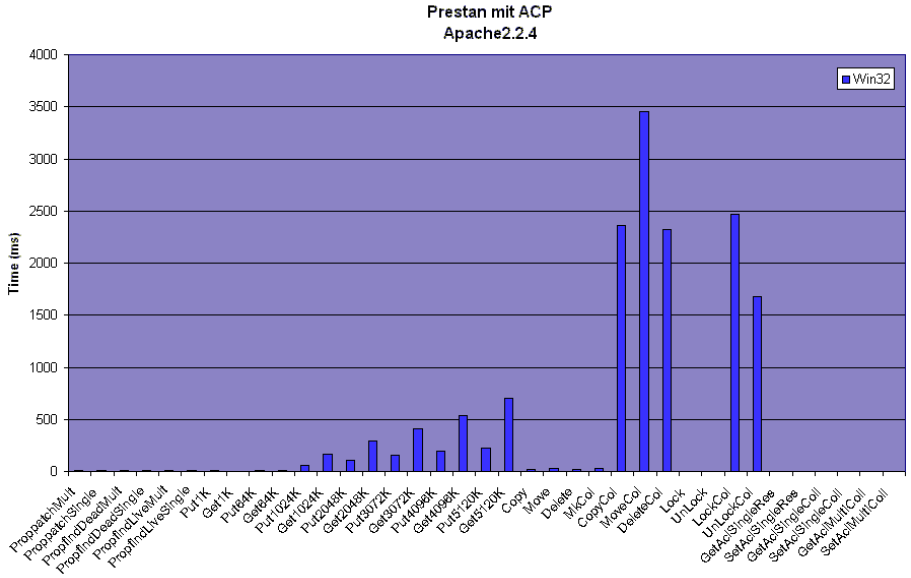

Abbildung B.13: Prestan mit ACP: Apache2.2.4 Win32

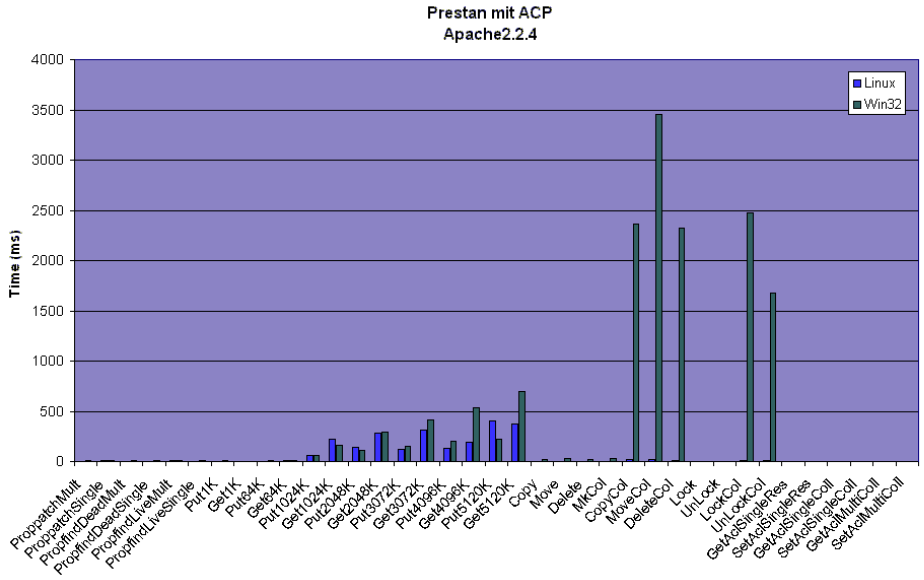

Abbildung B.14: Prestan mit ACP: Apache2.2.4 Linux/Win32

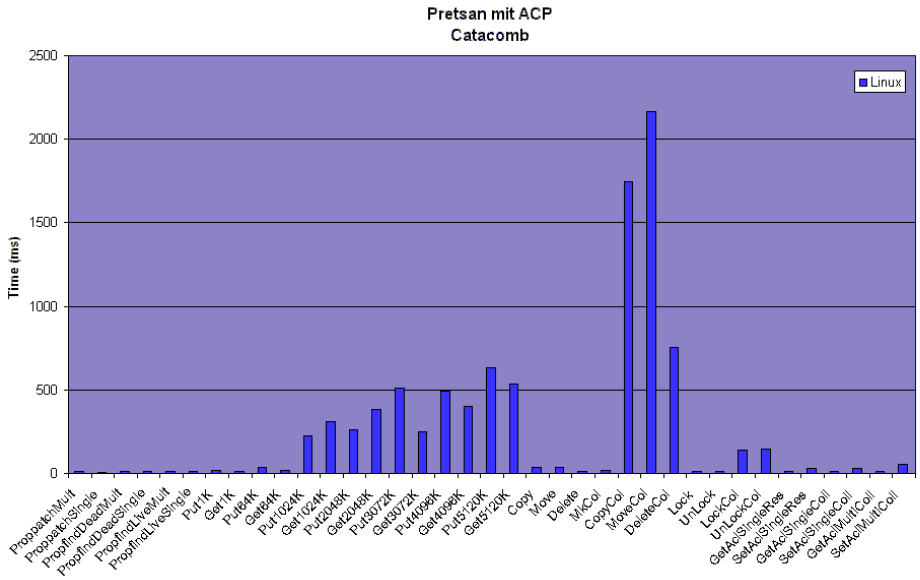

Abbildung B.15: Prestan mit ACP: Catacomb Linux

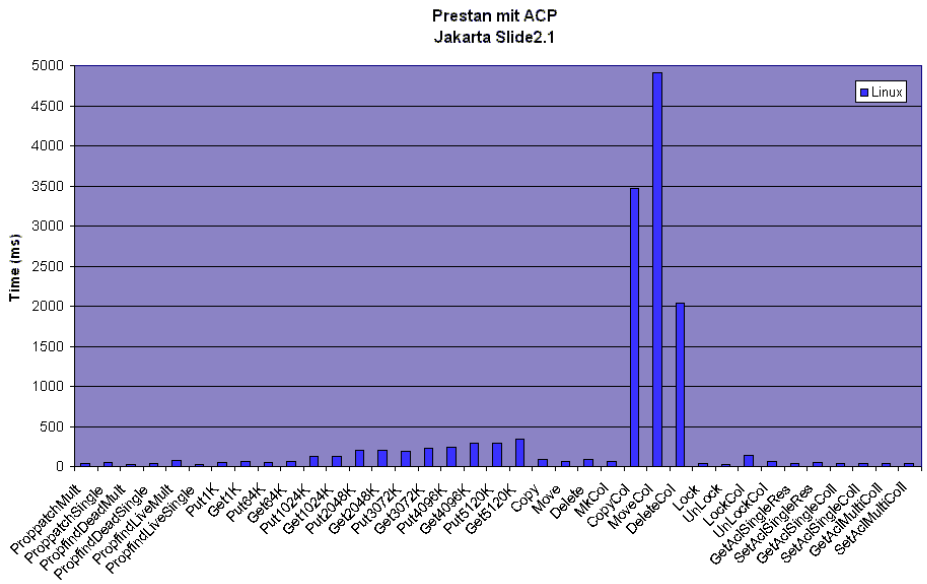

Abbildung B.16: Prestan mit ACP: Jakarta Slide2.1 Linux

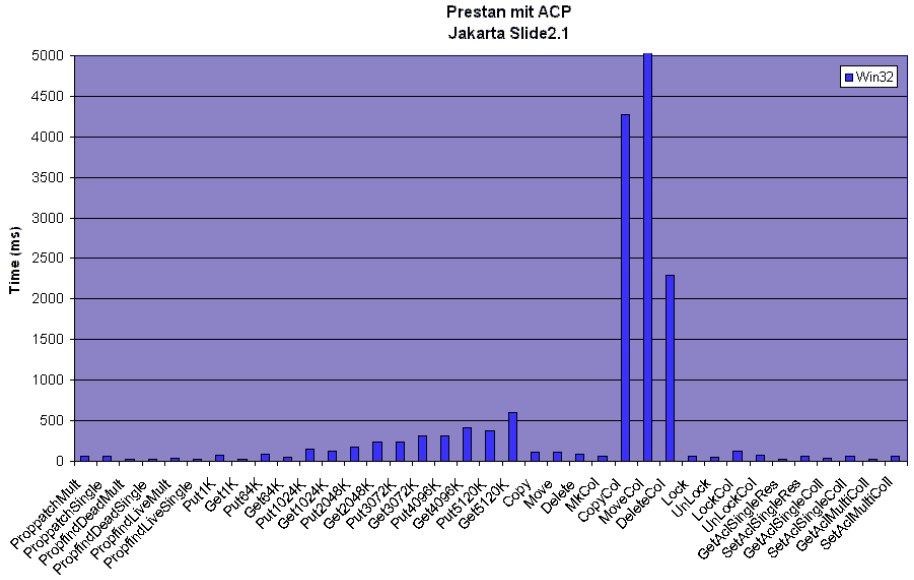

Abbildung B.17: Prestan mit ACP: Jakarta Slide2.1 Win32

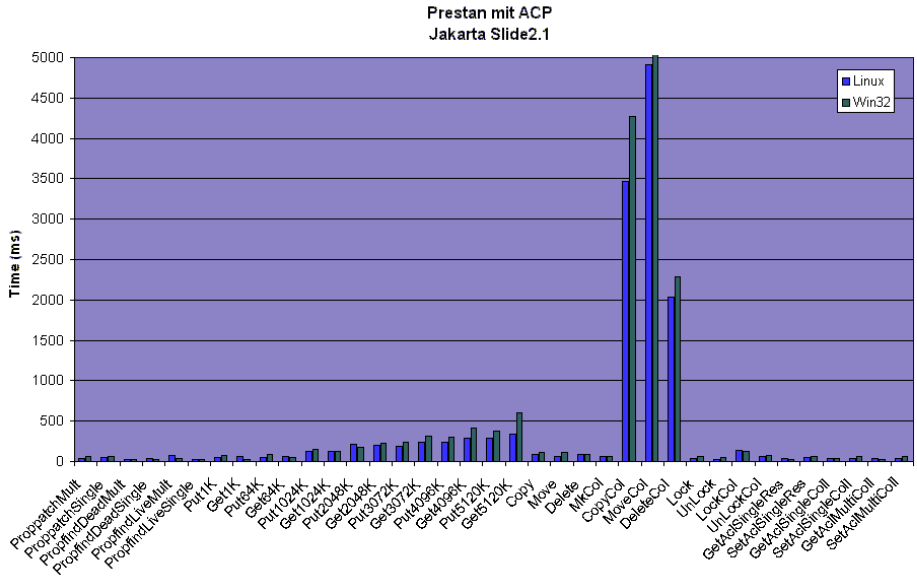

Abbildung B.18: Prestan mit ACP: Jakarta Slide2.1 Linux/Win32

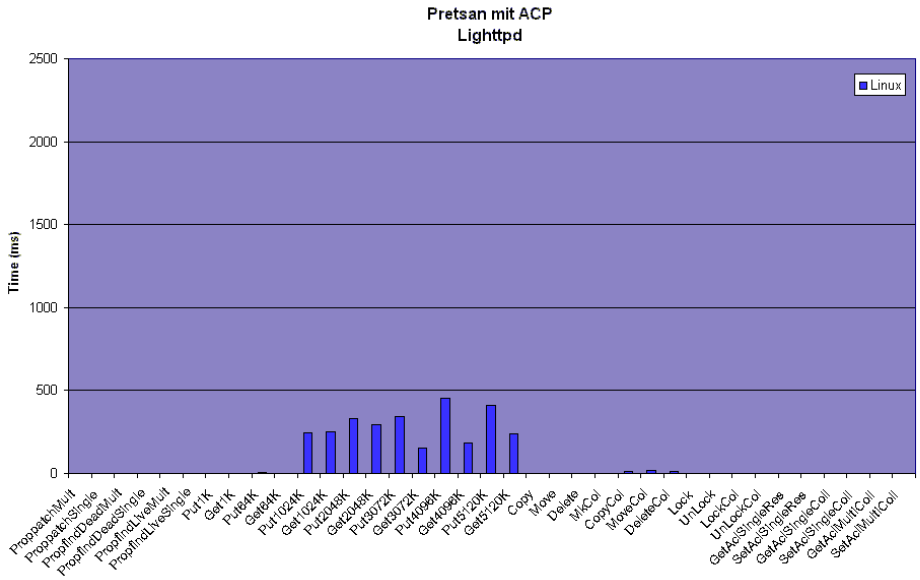

Abbildung B.19: Prestan mit ACP: Lighttpd1.4 Linux

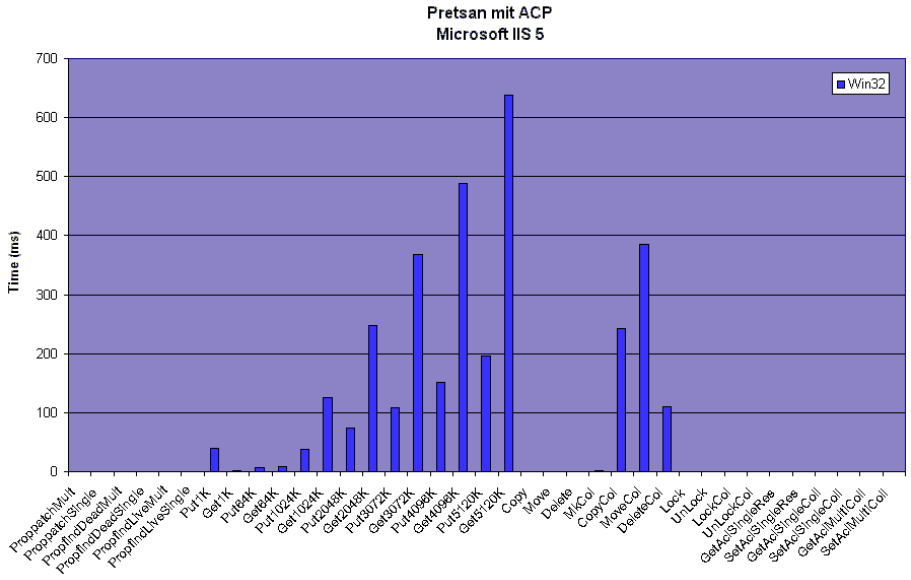

Abbildung B.20: Prestan mit ACP: Microsoft IIS5.1 Win32

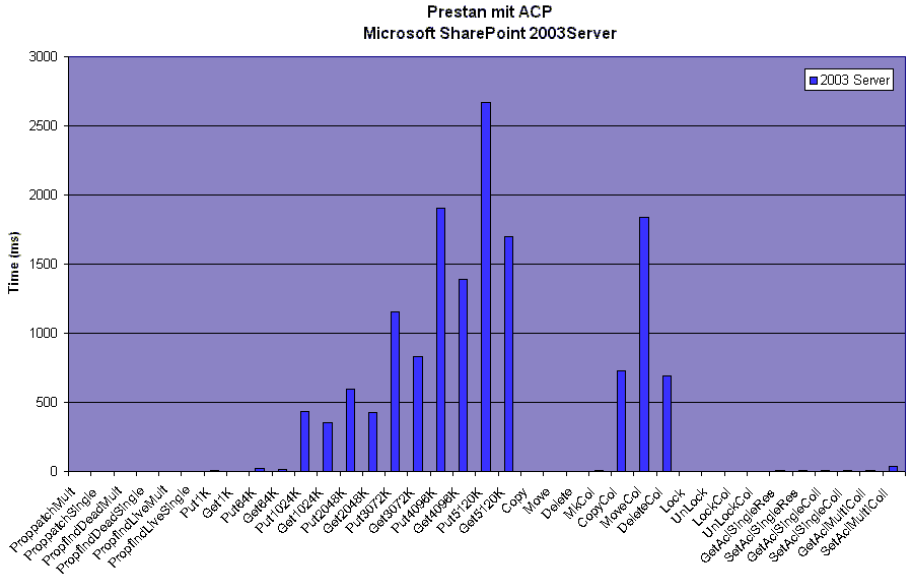

Abbildung B.21: Prestan mit ACP: SharePoint 2003 Server

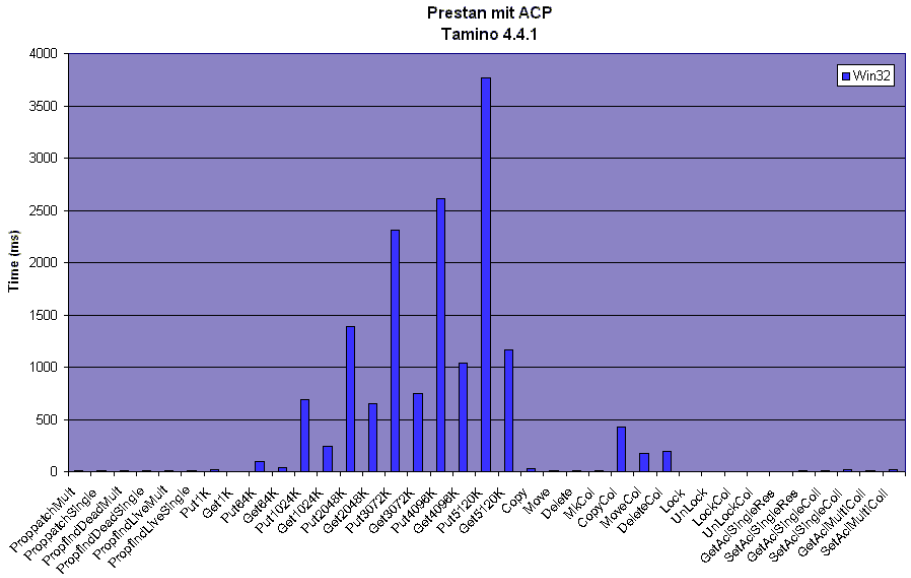

Abbildung B.22: Prestan mit ACP: Tamino4.4.1 Win32

Anhang C

Prestan - Wireshark Ergebnisse

```
OPTIONS /slide/files/davtest/acl-test/ HTTP/1.1
Host: 192.168.79.135:8080
User-Agent: davtest/0.1.0 neon/0.24.0-dev
Connection: TE
TE: trailers
Depth: 0
Content-Length: 0
Authorization: Basic cm9vdDpyb290
X-Prestan: (null): 2 (acl)

.
```

Listing C.1: Wireshark: OPTIONS Methode - Request

```
HTTP/1.1 200 OK
Pragma: No-cache
Cache-Control: no-cache
Expires: Thu, 01 Jan 1970 01:00:00 CET
Set-Cookie: JSESSIONID=54764B6ABBDC2674A96D953CE0F5B2B2; Path=/slide
DAV: 1, 2, slide, access-control, binding
DAV: version-control, version-history, checkout-in-place
DAV: workspace, working-resource, update, label
Allow: REPORT, CONNECT, BIND, MOVE, COPY, HEAD, TRACE,
POST, SEARCH, GET, DELETE, REBIND, ACL, PROPFIND,
UNBIND, LOCK, OPTIONS, PUT, PROPPATCH, UNLOCK
DASL: <DAV:basicsearch>
MS-Author-Via: DAV
Content-Length: 0
Date: Thu, 05 Jul 2007 18:19:33 GMT
Server: Apache-Coyote/1.1

.
```

Listing C.2: Wireshark: OPTIONS Methode - Response

```
PROPFIND /slide/files/davtest/acl-test/aclsingleres HTTP/1.1
Host: 192.168.79.135:8080
User-Agent: davtest/0.1.0 neon/0.24.0-dev
Connection: TE
TE: trailers
Depth: 0
Content-Length: 114
Authorization: Basic cm9vdDpyb290
X-Prestan: (null): 2 (acl)


<?xml version="1.0" encoding="utf-8"?>
<propfind xmlns='DAV:'>
 <prop>
  <acl xmlns='DAV:'/>
 </prop>
</propfind>

.
```

Listing C.3: Wireshark: Get ACL - Request - single Ressource

```
HTTP/1.1 207 Multi-Status
Pragma: No-cache
Cache-Control: no-cache
Expires: Thu, 01 Jan 1970 01:00:00 CET
Set-Cookie: JSESSIONID=B832B35263BF0A8DCC67959C89EF23B9; Path=/slide
Content-Type: text/xml;charset=UTF-8
Transfer-Encoding: chunked
Date: Thu, 05 Jul 2007 18:19:33 GMT
Server: Apache-Coyote/1.1


<?xml version="1.0" encoding="UTF-8"?>
<D:multistatus xmlns:D="DAV:">
<D:response>
 <D:href>/slide/files/davtest/acl-test/aclsingleres</D:href>
  <D:propstat>
   <D:prop>
    <D:acl>
      <D:ace>
       <D:principal>
        <D:unauthenticated />
       </D:principal>
       <D:grant>
        <D:privilege>
         <D:all />
        </D:privilege>
       </D:grant>
       <D:inherited>
        <D:href>/slide/files</D:href>
       </D:inherited>
      </D:ace>
      <D:ace>
       <D:principal>
        <D:href>/slide/roles/user</D:href>
       </D:principal>
       <D:grant>
        <D:privilege>
         <D:write />
        </D:privilege>
       </D:grant>
       <D:inherited>
        <D:href>/slide/files</D:href>
       </D:inherited>
      </D:ace>
      <D:ace>
       <D:principal>
        <D:property>
         <D:owner />
        </D:property>
       </D:principal>
       <D:grant>
        <D:privilege>
         <D:read-acl />
        </D:privilege>
       </D:grant>
       <D:inherited>
        <D:href>/slide/files</D:href>
       </D:inherited>
      </D:ace>
      <D:ace>
       <D:principal>
        <D:href>/slide/roles/root</D:href>
       </D:principal>
       <D:grant>
        <D:privilege>
         <D:all />
        </D:privilege>
       </D:grant>
       <D:inherited>
        <D:href>/slide/</D:href>
       </D:inherited>
      </D:ace>
      <D:ace>
       <D:principal>
        <D:all />
       </D:principal>
       <D:deny>
        <D:privilege>
         <D:read-acl />
        </D:privilege>
        <D:privilege>
```

```
          <D:write-acl />
         </D:privilege>
         <D:privilege>
          <D:unlock />
         </D:privilege>
        </D:deny>
        <D:inherited>
         <D:href>/slide/</D:href>
        </D:inherited>
       </D:ace>
       <D:ace>
        <D:principal>
         <D:all />
        </D:principal>
        <D:grant>
         <D:privilege>
          <D:read />
         </D:privilege>
        </D:grant>
        <D:inherited>
         <D:href>/slide/</D:href>
        </D:inherited>
       </D:ace>
     </D:acl>
    </D:prop>
    <D:status>HTTP/1.1 200 OK</D:status>
   </D:propstat>
</D:response>
</D:multistatus>

.
```

Listing C.4: Wireshark: Get ACL - Response - single Ressource

```
ACL /slide/files/davtest/acl-test/aclsingleres HTTP/1.1
Host: 192.168.79.135:8080
User-Agent: davtest/0.1.0 neon/0.24.0-dev
Connection: TE
TE: trailers
Depth: 0
Content-Length: 529
Authorization: Basic cm9vdDpyb290
X-Prestan: (null): 2 (acl)


<?xml version="1.0" encoding="utf-8"?>
<acl xmlns='DAV:'>
 <ace>
  <principal>
   <href>http://www.example.com/users/esedlar</href>
  </principal>
  <grant>
   <privilege><read/></privilege>
   <privilege><write/></privilege>
  </grant>
 </ace>
 <ace>
  <principal>
   <property><owner/></property>
  </principal>
  <grant>
   <privilege><read-acl/></privilege>
   <privilege><write-acl/></privilege>
  </grant>
 </ace>
 <ace>
  <principal><all/></principal>
  <grant>
   <privilege><read/></privilege>
  </grant>
 </ace>
</acl>

.
```

Listing C.5: Wireshark: Set ACL - Request - single Ressource

```
HTTP/1.1 200 OK
Pragma: No-cache
Cache-Control: no-cache
Expires: Thu, 01 Jan 1970 01:00:00 CET
```

```
Set-Cookie: JSESSIONID=BB7C6F6E022E54C168178B9CAA34E443; Path=/slide
Content-Length: 0
Date: Thu, 05 Jul 2007 18:19:34 GMT
Server: Apache-Coyote/1.1

.
```

Listing C.6: Wireshark: Set ACL - Response - single Ressource

```
PROPFIND /slide/files/davtest/acl-test/aclsinglecoll/ HTTP/1.1
Host: 192.168.79.135:8080
User-Agent: davtest/0.1.0 neon/0.24.0-dev
Connection: TE
TE: trailers
Depth: 0
Content-Length: 114
Authorization: Basic cm9vdDpyb290
X-Prestan: (null): 2 (acl)


<?xml version="1.0" encoding="utf-8"?>
<propfind xmlns='DAV:'>
 <prop>
  <acl xmlns='DAV:'/>
 </prop>
</propfind>

.
```

Listing C.7: Wireshark: Get ACL - Request - single Collection

```
HTTP/1.1 207 Multi-Status
Pragma: No-cache
Cache-Control: no-cache
Expires: Thu, 01 Jan 1970 01:00:00 CET
Set-Cookie: JSESSIONID=B832B35263BF0A8DCC67959C89EF23B9; Path=/slide
Content-Type: text/xml;charset=UTF-8
Transfer-Encoding: chunked
Date: Thu, 05 Jul 2007 18:19:33 GMT
Server: Apache-Coyote/1.1


<?xml version="1.0" encoding="UTF-8"?>
<D:multistatus xmlns:D="DAV:">
<D:response>
 <D:href>/slide/files/davtest/acl-test/aclsinglecoll</D:href>
  <D:propstat>
   <D:prop>
    <D:acl>
     <D:ace>
      <D:principal>
       <D:unauthenticated />
      </D:principal>
      <D:grant>
       <D:privilege>
        <D:all />
       </D:privilege>
      </D:grant>
      <D:inherited>
       <D:href>/slide/files</D:href>
      </D:inherited>
     </D:ace>
     <D:ace>
      <D:principal>
       <D:href>/slide/roles/user</D:href>
      </D:principal>
      <D:grant>
       <D:privilege>
        <D:write />
       </D:privilege>
      </D:grant>
      <D:inherited>
       <D:href>/slide/files</D:href>
      </D:inherited>
     </D:ace>
     <D:ace>
      <D:principal>
       <D:property>
        <D:owner />
       </D:property>
      </D:principal>
```

```
        <D:grant>
         <D:privilege>
          <D:read-acl />
         </D:privilege>
        </D:grant>
        <D:inherited>
         <D:href>/slide/files</D:href>
        </D:inherited>
       </D:ace>
       <D:ace>
        <D:principal>
         <D:href>/slide/roles/root</D:href>
        </D:principal>
        <D:grant>
         <D:privilege>
          <D:all />
         </D:privilege>
        </D:grant>
        <D:inherited>
         <D:href>/slide/</D:href>
        </D:inherited>
       </D:ace>
       <D:ace>
        <D:principal>
         <D:all />
        </D:principal>
        <D:deny>
         <D:privilege>
          <D:read-acl />
         </D:privilege>
         <D:privilege>
          <D:write-acl />
         </D:privilege>
         <D:privilege>
          <D:unlock />
         </D:privilege>
        </D:deny>
        <D:inherited>
         <D:href>/slide/</D:href>
        </D:inherited>
       </D:ace>
       <D:ace>
        <D:principal>
         <D:all />
        </D:principal>
        <D:grant>
         <D:privilege>
          <D:read />
         </D:privilege>
        </D:grant>
        <D:inherited>
         <D:href>/slide/</D:href>
        </D:inherited>
       </D:ace>
     </D:acl>
    </D:prop>
    <D:status>HTTP/1.1 200 OK</D:status>
   </D:propstat>
</D:response>
</D:multistatus>

.
```

Listing C.8: Wireshark: Get ACL - Response - single Collection

```
ACL /slide/files/davtest/acl-test/aclsinglecoll/ HTTP/1.1
Host: 192.168.79.135:8080
User-Agent: davtest/0.1.0 neon/0.24.0-dev
Connection: TE
TE: trailers
Depth: 0
Content-Length: 529
Authorization: Basic cm9vdDpyb290
X-Prestan: (null): 2 (acl)


<?xml version="1.0" encoding="utf-8"?>
<acl xmlns='DAV:'>
 <ace>
  <principal>
   <href>http://www.example.com/users/esedlar</href>
  </principal>
  <grant>
   <privilege><read/></privilege>
```

```
    <privilege><write/></privilege>
   </grant>
  </ace>
  <ace>
   <principal>
    <property><owner/></property>
   </principal>
   <grant>
    <privilege><read-acl/></privilege>
    <privilege><write-acl/></privilege>
   </grant>
  </ace>
  <ace>
   <principal><all/></principal>
   <grant>
    <privilege><read/></privilege>
   </grant>
  </ace>
</acl>

.
```

Listing C.9: Wireshark: Set ACL - Request - single Collection

```
HTTP/1.1 200 OK
Pragma: No-cache
Cache-Control: no-cache
Expires: Thu, 01 Jan 1970 01:00:00 CET
Set-Cookie: JSESSIONID=3F24ED604D032C2D0A061C842A49AB97; Path=/slide
Content-Length: 0
Date: Thu, 05 Jul 2007 18:19:34 GMT
Server: Apache-Coyote/1.1

.
```

Listing C.10: Wireshark: Set ACL - Request - single Collection

```
PROPFIND /slide/files/davtest/acl-test/aclmulticoll/ HTTP/1.1
Host: 192.168.79.135:8080
User-Agent: davtest/0.1.0 neon/0.24.0-dev
Connection: TE
TE: trailers
Depth: 0
Content-Length: 114
Authorization: Basic cm9vdDpyb290
X-Prestan: (null): 2 (acl)


<?xml version="1.0" encoding="utf-8"?>
<propfind xmlns='DAV:'>
 <prop>
  <acl xmlns='DAV:'/>
 </prop>
</propfind>

.
```

Listing C.11: Wireshark: Get ACL - Request - multi Collection

```
HTTP/1.1 207 Multi-Status
Pragma: No-cache
Cache-Control: no-cache
Expires: Thu, 01 Jan 1970 01:00:00 CET
Set-Cookie: JSESSIONID=27FF172AFA2B247F488EA21227F3C3E0; Path=/slide
Content-Type: text/xml;charset=UTF-8
Transfer-Encoding: chunked
Date: Thu, 05 Jul 2007 18:19:35 GMT
Server: Apache-Coyote/1.1


<?xml version="1.0" encoding="UTF-8"?>
<D:multistatus xmlns:D="DAV:">
<D:response>
 <D:href>/slide/files/davtest/acl-test/aclmulticoll</D:href>
  <D:propstat>
   <D:prop>
    <D:acl>
     <D:ace>
```

```
        <D:principal>
         <D:unauthenticated />
        </D:principal>
        <D:grant>
         <D:privilege>
          <D:all />
         </D:privilege>
        </D:grant>
        <D:inherited>
         <D:href>/slide/files</D:href>
        </D:inherited>
       </D:ace>
       <D:ace>
        <D:principal>
         <D:href>/slide/roles/user</D:href>
        </D:principal>
        <D:grant>
         <D:privilege>
          <D:write />
         </D:privilege>
        </D:grant>
        <D:inherited>
         <D:href>/slide/files</D:href>
        </D:inherited>
       </D:ace>
       <D:ace>
        <D:principal>
         <D:property>
          <D:owner />
         </D:property>
        </D:principal>
        <D:grant>
         <D:privilege>
          <D:read-acl />
         </D:privilege>
        </D:grant>
        <D:inherited>
         <D:href>/slide/files</D:href>
        </D:inherited>
       </D:ace>
       <D:ace>
        <D:principal>
         <D:href>/slide/roles/root</D:href>
        </D:principal>
        <D:grant>
         <D:privilege>
          <D:all />
         </D:privilege>
        </D:grant>
        <D:inherited>
         <D:href>/slide/</D:href>
        </D:inherited>
       </D:ace>
       <D:ace>
        <D:principal>
         <D:all />
        </D:principal>
        <D:deny>
         <D:privilege>
          <D:read-acl />
         </D:privilege>
         <D:privilege>
          <D:write-acl />
         </D:privilege>
         <D:privilege>
          <D:unlock />
         </D:privilege>
        </D:deny>
        <D:inherited>
         <D:href>/slide/</D:href>
        </D:inherited>
       </D:ace>
       <D:ace>
        <D:principal>
         <D:all />
        </D:principal>
        <D:grant>
         <D:privilege>
          <D:read />
         </D:privilege>
        </D:grant>
        <D:inherited>
         <D:href>/slide/</D:href>
        </D:inherited>
       </D:ace>
      </D:acl>
     </D:prop>
```

```
    <D:status>HTTP/1.1 200 OK</D:status>
  </D:propstat>
</D:response>
</D:multistatus>

.
```

Listing C.12: Wireshark: Get ACL - Response - multi Collection

```
DELETE /slide/files/davtest/acl-test/ HTTP/1.1
Host: 192.168.79.135:8080
User-Agent: davtest/0.1.0 neon/0.24.0-dev
Connection: TE
TE: trailers
Depth: 0
Content-Length: 0
Authorization: Basic cm9vdDpyb290
X-Prestan: (null): 2 (acl)

.
```

Listing C.13: Wireshark: DELETE Methode - Request

```
HTTP/1.1 204 No Content
Pragma: No-cache
Cache-Control: no-cache
Expires: Thu, 01 Jan 1970 01:00:00 CET
Set-Cookie: JSESSIONID=553E3D2567D34384B1C742DB591D91D9; Path=/slide
Date: Thu, 05 Jul 2007 18:19:35 GMT
Server: Apache-Coyote/1.1

.
```

Listing C.14: Wireshark: DELETE Methode - Response

Anhang D

Inhalt der CD-ROM

Dieser Diplomarbeit wurde eine CD-ROM beigelegt, die folgende Verzeichnisse beinhaltet:

Quellcode
Im Quellcodeverzeichnis sind die aktuellen Versionen von Litmus und Prestan enthalten.

Quellen
In diesem Verzeichnis befinden sich einige Dokumente, die für die Arbeit benutzt wurden.

Testergebnisse
Hier befinden sich alle Testergebnisse, die von Litmus und Prestan erzeugt worden sind.

Diplomarbeit
In diesem Verzeichnis befindet sich die Diplomarbeit im PDF-Format. Zusätzlich sind hier noch andere Dateien (Bilder, Tabellen ...) zugefügt, die für die Arbeit entwickelt wurden.

Dokumnetation
Hier wird die generierte Code-Dokumentation der ACP- und XML-Erweiterung hinterlegt.