Studienarbeit

Caching und Zugriffsprotokollierung im World Wide Web

Eine einführende Untersuchung

Betreuer:
Prof. Dr. Joachim W. Schmidt
Claudia Niederée

Jan Stövesand
Rückertstr. 27
22089 Hamburg

Universität Hamburg
Fachbereich Informatik
Datenbanken und Informationssysteme

Inhaltsverzeichnis

1 Einleitung und Motivation **2**
- 1.1 Einführung in die Thematik und Terminologie . . . 2
- 1.2 Aufgabenstellung . . . 4
- 1.3 Struktur der Arbeit . . . 5

2 Caching und Zugriffsprotokollierung **7**
- 2.1 Caching-Verfahren . . . 7
  - 2.1.1 Auswahlverfahren der Browser . . . 7
  - 2.1.2 Mehrstufiges Caching . . . 9
  - 2.1.3 Verfahren der Zwischenspeicherung . . . 12
  - 2.1.4 Caching-Kontrolle im HTTP-Protokoll . . . 14
- 2.2 Caching-Topologien . . . 15
  - 2.2.1 Kaskadierte Caches . . . 15
  - 2.2.2 Hierarchische Caches . . . 15
- 2.3 Zugriffsprotokollierung . . . 16
  - 2.3.1 Standard-Protokolldateien und -Formate . . . 17
  - 2.3.2 Erweiterte Protokolldateien . . . 18
  - 2.3.3 Benutzerdefinierte Protokolldateien . . . 19
  - 2.3.4 Zustandsprotokollierung mit Magic-Cookies . . . 19

3 Auswirkungen von Caching-Verfahren **21**
- 3.1 Verfälschung der Zugriffszahlen auf Web-Dokumente . . . 21
  - 3.1.1 Modellbildung . . . 22
  - 3.1.2 Benachrichtigung der Server durch die Caches . . . 22
  - 3.1.3 Verwendung nicht zwischenspeicherbarer Web-Dokumente . . . 23
- 3.2 Versionsabgleich von Web-Seiten . . . 23
  - 3.2.1 Angabe eines Verfallsdatum . . . 23
  - 3.2.2 Angabe des Datums der letzten Änderung . . . 24
  - 3.2.3 Überprüfung der Aktualität eines Web-Dokuments . . . 24
- 3.3 Verwendung von IP-Maskierung . . . 25

4 Korrektur von Zugriffszahlen **27**
- 4.1 Idee des Verfahrens . . . 27
- 4.2 Umsetzung des Verfahrens . . . 28
- 4.3 Probleme des Verfahrens . . . 30
  - 4.3.1 Der Image-Off Faktor . . . 30

4.3.2 Probleme bei der Benutzung von Frames 32
4.3.3 Unnötiges Datenaufkommen 33
4.4 Exemplarische Messungen 33
4.4.1 Charakteristika der untersuchten Web-Server und Web-Seiten 33
4.4.2 Messungen der ersten Versuchsanordnung 34
4.4.3 Messungen der zweiten Versuchsanordnung 35
4.4.4 Bewertung der Messungen 36
4.5 Bewertung des Verfahrens 36
4.6 Vorschlag eines Alternativverfahrens 38

5 Bewertung und Ausblick **41**

Literaturverzeichnis **43**

Abbildungsverzeichnis

1.1 Direkte Anforderung einer Web-Seite 3
1.2 Anforderung einer Web-Seite mit Cache 4

2.1 Browser mit Kenntnis der verfügbaren Caches 8
2.2 Browser mit stellvertretendem Cache (Stellvertreter) 8
2.3 Verwendung eines Firewalls zum Verbinden eines internen Netzwerkes mit dem Internet 9
2.4 Verschiedene Ebenen von stellvertretenden Caches 11
2.5 Kaskadierte Caches 15
2.6 Hierarchische Caches 16

3.1 Verhalten eines Web-Servers bei einer bedingten Anforderung 24
3.2 IP-Maskierung durch einen Cache 26

4.1 Verlauf der Anforderung einer Web-Seite mit eingebettetem nicht zwischenspeicherbaren Web-Dokument 28
4.2 Darstellung der Demonstrations-Homepage mit eingebetteter Fliege 30
4.3 Demonstrations-Homepage ohne automatisches Nachladen 31
4.4 Zugriffe auf die Homepage und die Fliege des DBIS-Servers 35
4.5 Zugriffe auf die Homepage und die Fliege des Zeitungs-Servers 36
4.6 Verhältnisse der Zugriffe auf die jeweilige Fliege und Homepage auf den Servern 37
4.7 Zugriffe auf zwei Web-Seiten des Zeitungs-Servers und deren Fliegen 38
4.8 Verhältnisse der Zugriffe bei der zweiten Versuchsanordnung 39

Kapitel 1

Einleitung und Motivation

Zu Beginn dieser Arbeit erfolgt eine kurze Einführung in das Internet. Durch diesen kurzen Überblick soll eine einheitliche Terminologie festgelegt werden, auf der die folgenden Kapitel aufbauen. Außerdem wird die Aufgabenstellung dieser Arbeit vorgestellt.

1.1 Einführung in die Thematik und Terminologie

Das Internet ist Anfang der 60'er Jahre aus einem Forschungsprojekt des Verteidigungsministeriums der USA (*Department of Defense*, DoD) entstanden. Die mit diesem Projekt beauftragte Abteilung des DoD war die *Advanced Research Projects Argency* (ARPA), weswegen das Netz anfänglich noch den Namen ARPAnet trug. 1970 wurden das neue Netzwerk zwischen vier Universitäten von der ARPA installiert. Innerhalb der nächsten zwei Jahre wuchs das ARPAnet auf über 40 Knoten an. Es wurden Dienste entwickelt, die unter anderem

- das Versenden von Textnachrichten (*Electronic Mail*),
- das Arbeiten auf entfernt stehenden Rechner (*Remote Login*) und
- den Austausch von Dateien ermöglichen [ea95].

Mittlerweile ist aus dem kleinen Forschungsnetz ein weltumspannendes Netzwerk geworden, das *Internet*. Neben den bereits erwähnten Diensten *Elektronische Post* (Email), Fernsteuerung von Rechnern und dem Austausch von Dateien, sind unter anderem noch Dienste für ein *Schwarzes Brett* (NEWS) [KL86] und für die Übertragung von Hypertexten hinzugekommen.
Für jeden dieser Dienste wurde ein Protokoll eingeführt, das die Form des Datenaustauschs für die jeweiligen Dienste regelt. So entstand das *Simple Mail Transfer Protokoll* (SMTP) [Pos85b] für die Elektronische Post, das *File Transfer Protokoll* (FTP) [Pos85a] für den Austausch von Dateien und das *Hypertext Transfer Protokoll* (HTTP) [BLFF95] für die Übertragung von Hypertexten.
Letzteres bildet die Grundlage für das World Wide Web (WWW), welches in den letzten Jahren erheblich an Bedeutung gewonnen hat. Das WWW ist die globale Sammlung von Hypertexten, die von sogenannten Web-Servern abgerufen werden können. Diese Hypertexte können Querverweise (*Links*) auf andere Hypertexte enthalten, die sich auf einem beliebigen Web-Server im Internet befinden können. Im Kontext des WWW werden diese

Hypertexte als Web-Seiten bezeichnet. Zusätzlich können noch Bilder, Klänge oder ähnliche *Multimedia-Elemente* in den Hypertext eingebettet werden. Die Übertragung einer Web-Seite im WWW erfolgt in einem oder mehreren Web-Dokumenten. Der textuelle Teil inklusive Formatierungen (fett, kursiv, unterstrichen, ...) sowie das Layout werden zuerst in einem separaten *Web-Dokument* übertragen. Die Kodierung dieser Information, also der Text, die Querverweise, sowie die Positionen und Adressen der eingebetteten Multimedia-Elemente, erfolgt in einer Seitenbeschreibungssprache, der Hypertext Markup Language (HTML) [Con95a, Con96a, Con96b]. Die Multimedia-Elemente können in separaten Web-Dokumenten von den Web-Servern nachgeladen werden. Eine Web-Seite ist somit eine Aggregation aus einem oder mehreren Web-Dokumenten. Geladen, zusammengesetzt und angezeigt werden die Web-Seiten von sogenannten Web-Browsern, die außerdem Navigationshilfen, z.B. zum Verfolgen eines Links, zur Verfügung stellen.
Jedes Web-Dokument hat eine eindeutige Bezeichnung, damit man es im WWW referenzieren kann. Dies ist die sogenannte URL (Uniform Resource Locator), welche die folgende Form hat:

```
<Protokoll>://<server>[:<Port>]/<Dokumentname mit Pfad>
```

(z.B. http://www.uni-hamburg.de:80/index.html)
Die URL enthält das Protokoll, die IP-Adresse des Web-Servers, sowie den Namen des Web-Dokuments mit Pfadangabe. Wenn ein Web-Dokument von einem Browser mittels der URL geladen werden soll, wird als erstes die Adresse des Web-Servers ausgewertet und versucht eine Verbindung mit dem spezifizierten Port aufzubauen. Dabei wird das dem WWW zugrundeliegende *HTTP-Protkoll* [BLFF95] verwendet. Gelingt der Verbindungsaufbau, sendet der Browser die eigentliche Anforderung in Form eines *HTTP- Requests* an den Server und wartet auf eine Antwort. Ist das angeforderte Web-Dokument auf dem Server vorhanden, wird es an den Browser zurückgeschickt. Abbildung 1.1 verdeutlicht diese Schritte.

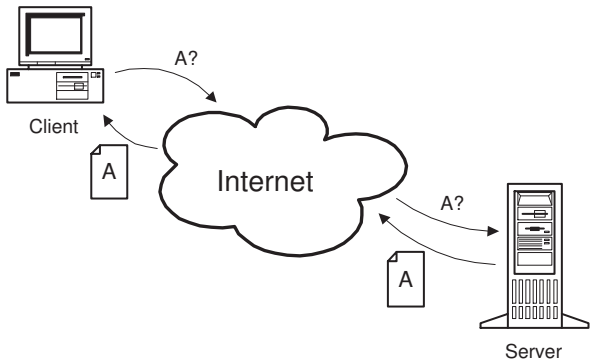

Abbildung 1.1: Direkte Anforderung einer Web-Seite

Für jedes angeforderte Web-Dokument muß eine Verbindung zum Web-Server aufgebaut, die Anforderung gesendet, auf die Antwort des Web-Servers gewartet und schließlich die Verbindung wieder abgebaut werden. Seit der verstärkten Nutzung des WWW führt dies

zu einer erheblichen Belastung des Internets und hat negative Auswirkungen auf Geschwindigkeit und Antwortzeit.
Um dies zu kompensieren, wurde ein Prinzip übernommen, welches in der technischen Informatik seit langem Anwendung findet: Um die Geschwindigkeit des Datenaustausches zwischen Prozessor und Hauptspeicher zu beschleunigen, wird ein Cache zwischen Prozessor und Hauptspeicher gesetzt, welcher verwendete Daten zwischenspeichert. Fordert der Prozessor nun den Inhalt einer Speicherzelle an, prüft der Cache zunächst, ob eine lokale Kopie vorhanden ist und gibt diese dann direkt an den Prozessor, ohne die Daten über den Bus vom langsameren Hauptspeicher erneut zu holen.
Im Internet werden die Wartezeiten vermindert und die Geschwindigkeit erhöht, indem man von oft benötigten Web-Dokumenten eine Kopie auf *Cache-Rechnern* (im weiteren einfach *Cache* genannt) zwischenspeichert. Um einen Beschleunigungs-Effekt zu erreichen müssen diese Caches im Internet so plaziert sein, daß sie näher an den Browsern liegen als die Server. Es bietet sich zum Beispiel an, den Rechner eines Internetproviders als Cache zu verwenden, da alle Benutzer über diesen Rechner mit dem Internet verbunden sind und alle Anforderungen somit über diesen Rechner laufen müssen. Eine Anforderung muß dann nicht die weitere Strecke zum Server zurücklegen, wenn die angeforderte Seite sich als Kopie auf dem Cache befindet. Dies hat außerdem noch zur Folge, das weniger Teilstrecken im Netz beansprucht werden, was sich generell positiv auf die Leistung im Internet auswirkt. Abbildung 1.2 zeigt, wie eine Web-Seite von einem Cache bezogen wird, ohne eine direkte Verbindung zum Server aufzubauen.

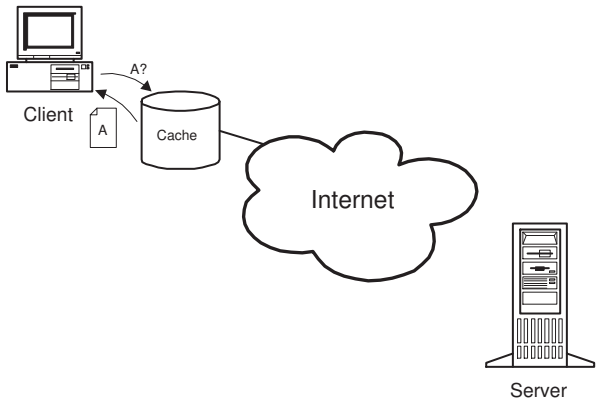

Abbildung 1.2: Anforderung einer Web-Seite mit Cache

1.2 Aufgabenstellung

Die vereinfachte Benutzung des Internets mittels des WWW hat zu stark ansteigenden Benutzerzahlen geführt. Mit dieser Entwicklung ging eine verstärkte Kommerzialisierung einher, da das WWW von Firmen als neues Werbemedium entdeckt wurde. Die Akzeptanz einer solchen Online-Präsenz kann anhand der auf den Web- Servern geführten Protokolldateien analysiert werden. Das Caching, welches benötigt wird, um die Geschwindigkeit

und das Antwortverhalten des WWW zu verbessern, verfälscht aber diese Protokolldateien und es bedarf Korrekturverfahren, die diesen Verfälschungen entgegenwirken bzw. diese kompensieren. Da diese Korrekturverfahren auf der Caching-Technologie aufbauen, wird in dieser Arbeit zunächst ein Überblick über den heutigen Stand dieser Technologie gegeben, und es werden damit verbundene Probleme vorgestellt. Ebenso werden die Protokolldateien beschrieben, die Basis für die Analysen sind.
Diese Übersicht bildet die Grundlage für das Verständnis der Korrekturverfahren, welche den Schwerpunkt dieser Arbeit darstellen. Mögliche Korrekturverfahren werden untersucht und es wird der Frage nachgegangen, welche dieser Verfahren praktikabel bzw. sinnvoll sind.

1.3 Struktur der Arbeit

Die Grundprinzipien des Caching, Verfahren der Speicherung, Cache-Verbunde, Zugriffsprotokollierung, etc. werden in Kapitel 2 erklärt.
Kapitel 3 befaßt sich mit den Folgen des Caching auf die Aktualität von Web-Dokumenten und damit; wie diese reduziert werden können. Weiterhin werden die Folgen für die Zugriffsprotokollierung untersucht.
In Kapitel 4 dieser Arbeit wird ein Verfahren vorgestellt, welches die Korrektur von Zugriffszahlen ermöglicht. Die Vor- und Nachteile dieses Verfahrens werden erläutert. In einer exemplarischen Anwendung auf zwei Web-Servern wird das Verfahren demonstriert und es wird eine erste Bewertung vorgenommen. Abschließend wird der Vorschlag für ein Alternativverfahren erarbeitet.
In Kapitel 5 wird eine Zusammenfassung über die in dieser Arbeit erzielten Ergebnisse gegeben.

Kapitel 2

Caching und Zugriffsprotokollierung im World Wide Web

Eine Untersuchung der Auswirkungen des Cachings auf die Zugriffsprotokollierung, die Kern dieser Arbeit ist, erfordert zunächst eine Auseinandersetzung mit den beiden relevanten Teilgebieten Zugriffsprotokollierung und Caching. In diesem Kapitel wird ein technischer Überblick über die verschiedenen Arten von Caches gegeben. Außerdem werden verschiedene Cache-Topologien vorgestellt. Um die Zugriffe auf einen Web-Server und einen Cache verfolgen zu können, werden diese protokolliert. Dies kann auf verschiedene Arten und in verschiedenen Formaten erfolgen. Ein Überblick über diese Protokolldateien und -formate wird im letzten Abschnitt gegeben.

2.1 Caching-Verfahren

Generell ist es die Aufgabe eines Caches, die Zugriffe auf Web-Dokumente innerhalb des WWW zu beschleunigen. Erreicht werden soll dies, indem Web-Dokumente auf den Caches als Kopie zwischengespeichert werden und somit bei einem erneuten Zugriff schneller verfügbar sind. Um den gewünschten Effekt zu erzielen, muß gewährleistet sein, daß der Zugriff auf den Cache schneller erfolgen kann, als der Zugriff auf den ursprünglichen Server. Wenn ein Browser ein bereits im Cache vorhandenes Dokument anfordert, kann dieses durch den Cache geliefert werden, und es liegt ein *Cache-Treffer* (*cache hit*) vor. Bei der Untersuchung von Caching-Verfahren kann zwischen unterschiedlichen Auswahlverfahren, Topologien und Ansätzen bei der Zwischenspeicherung unterschieden werden.

2.1.1 Auswahlverfahren der Browser

Bei der Auswahl, ob und welcher Cache vom Browser verwendet werden soll, gibt es zwei grundlegende Techniken. Sie unterscheiden sich darin, ob der Browser den oder die Caches implizit oder explizit wählt.

- **Explizite Auswahl**
  Bei dieser Technik wird dem Browser eine Liste mit Caches zur Verfügung gestellt.

Die Liste der Caches wird sequentiell abgearbeitet, bis ein Cache-Treffer erzielt wurde oder kein weiterer Cache mehr zur Verfügung steht. Nur wenn das Web-Dokument von keinem der Caches bezogen werden kann, wird die Anforderung zum Web-Server (siehe Abbildung 2.1) geschickt. Durch die explizite Auswahl der verwendeten Caches (*cache awareness*) besitzt der Browser die Kontrolle über die Herkunft der Web-Dokumente.

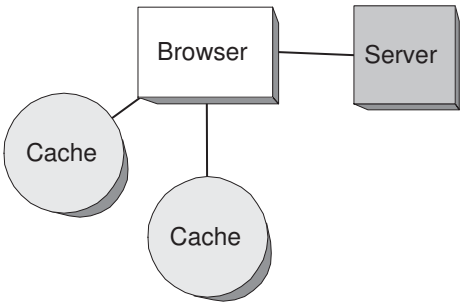

Abbildung 2.1: Browser mit Kenntnis der verfügbaren Caches

- **Implizite Auswahl**
  Bei der zweiten Technik wird der Browser an einen *stellvertretenden Cache* (*proxy*), im weiteren einfach *Stellvertreter* genannt, angeschlossen. Der Browser sendet jede Anforderung an seinen Stellvertreter und erhält das angeforderte Web-Dokument auch von diesem zurück. Der Stellvertreter fungiert also als Zwischenstation zwischen Browser und Web-Server (siehe Abbildung 2.2). Dabei kann der Browser nicht feststellen, ob das Dokument aus dem Speicher des Stellvertreters stammt oder ob dieser das Dokument vom Web-Server angefordert hat (*cache unawareness*).

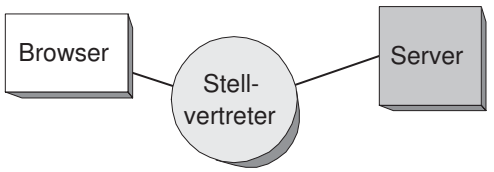

Abbildung 2.2: Browser mit stellvertretendem Cache (Stellvertreter)

Beide Techniken weisen Vor- und Nachteile auf, die die Verwendung der einen oder anderen Technik rechtfertigen. Die Konfiguration von Browsern mit explizitem Auswahlverfahren gestaltet sich etwas aufwendiger, da jeder zu verwendende Cache spezifiziert werden muß. Die Leistung einer solchen Konfiguration hängt in starkem Maße von einer leistungsorientierten Auswahlreihenfolge für die Caches ab. Hier liegt die Schwachstelle dieses Verfahrens, da man nur mit genauer Kenntnis der Hard- und Software der Caches Aussagen über deren Leistungsfähigkeit machen kann. Die Vorteile liegen zum einen in der Kontrolle der

Herkunft der Web-Dokumente, wodurch Aktualität sichergestellt werden kann, sowie in der Möglichkeit die Last auf mehrere Caches gezielt zu verteilen (*load balancing*). Letzteres ermöglicht die gezielte Vermeidung bzw. Umgehung von Netzengpässen [CDN+96].
Die Konfiguration von Browsern der zweiten Kategorie gestaltet sich dagegen sehr einfach, da nur der Stellvertreter anzugeben ist. Leistungsbestimmend bei dieser Art der Konfiguration ist die ausreichende Dimensionierung des Stellvertreters. Problematisch ist die geringe Fehlertoleranz, da bei Ausfall des Stellvertreters eine Benutzung des Internets nicht mehr möglich ist.
Diese zweite Technik wird von den meisten modernen Browsern unterstützt, da sie eine Möglichkeit zum Überwinden von *Firewalls* ist [CZ95]. Firewalls sind Rechner, die das öffentliche Internet mit einem privaten Netzwerk verbinden (siehe Abb. 2.3). Hierdurch können Sicherheitsrisiken vermindert und private Daten vor unzulässigem Zugriff geschützt werden. Firewalls bieten aber nicht nur den Schutz vor Eindringlingen, sondern können auch den Datenfluß zwischen dem privaten Netzwerk und dem Internet überwachen und einschränken. Außerdem können ganze Netzwerke durch Firewalls für das Internet verborgen werden, indem bei allen ausgehenden Paketen die Adresse des Absenders durch die Adresse des Firewalls ersetzt wird (*IP-Masquerading*, siehe Abschnitt 3.3). Da alle ausgehenden Verbindungen aus dem privaten Netzwerk über den Firewall laufen müssen, stellt dieser einen potentiellen Netzengpaß dar. Es hat sich daher als sinnvoll erwiesen, daß der Stellvertreter innerhalb des Firewalls auch die Aufgaben eines Caches übernimmt, um die Netzlast hier gezielt zu senken. Außerdem durchlaufen alle Web-Dokumente den Stellvertreter, weswegen dieser für die Übernahme von Caching-Aufgaben ohnehin prädestiniert ist.

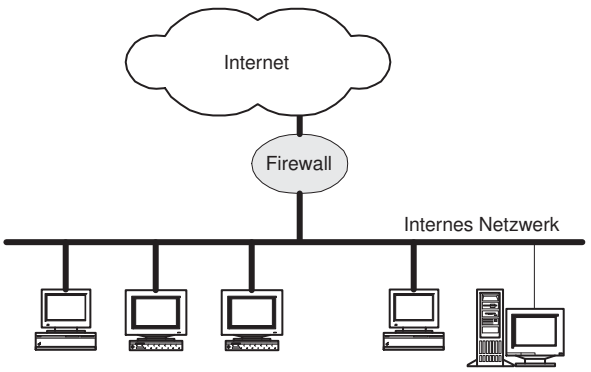

Abbildung 2.3: Verwendung eines Firewalls zum Verbinden eines internen Netzwerkes mit dem Internet

2.1.2 Mehrstufiges Caching

Die meistverwendeten Browser (Netscape Navigator, Microsoft Internet Explorer, NCSA Mosaic) verwenden eine Mischform der beiden im vorigen Abschnitt erläuterten Techniken. Dadurch entsteht ein mehrstufiges Auswahlverfahren, welches sich in eine interne und eine externe Stufe untergliedern läßt. Bei der internen Stufe wird zunächst der *lokale Cache*

konsultiert. Kann hier kein Cache-Treffer erzielt werden, wird die Anforderung an einen Stellvertreter weitergeleitet, wodurch die zweite, die externe Stufe eingeleitet wird. Dieses Auswahlverfahren ist zum einen explizit, da der Browser zuerst den internen Cache wählt und erst dann die Anforderung an einen externen weiterleitet. Somit ist zumindest bekannt, ob das angeforderte Web-Dokument vom lokalen Cache geliefert wurde. In der zweiten Stufe wird das implizite Auswahlverfahren benutzt. Der Browser kann über die Herkunft eines Web-Dokumentes also nur zwei Aussagen machen. Entweder es stammt aus dem internen Cache oder wurde von einem externen Rechner bezogen. Dabei kann es sich sowohl um einen externen Cache, als auch um einen Web-Server handeln.
Um die Leistung dieser Technik noch zu steigern, kann man das Prinzip der stellvertretenden Caches mehrfach anwenden. Einem Stellvertreter kann wiederum ein Stellvertreter zugeordnet werden. Kann kein Cache-Treffer erzielt werden, wird die Anforderung nicht an den Web-Server, sondern an den nächsten Stellvertreter gesendet. Dies kann auf beliebig viele Hierarchiestufen, welche die *regionalen und nationalen Caches* bilden, ausgedehnt werden. Durch verschiedene denkbare An- und Zuordnungen, dieser stellvertretenden Caches, entstehen unterschiedliche Caching-Topologien, welche in Abschnitt 2.2 eingehend erläutert werden. Abbildung 2.4 zeigt eine Anordnung und veranschaulicht die unterschiedliche Stellung der lokalen und regionalen/nationalen Caches, die im folgenden genauer beschrieben werden.

- **Lokale Caches**
  Lokale Caches sind im Web-Browser implementiert und speichern die Daten im Dateibereich des Benutzers. Hierdurch nehmen diese Caches eine Sonderposition unter den verschiedenen Arten von Caches ein, da sie durch die Bindung an einen Benutzer dessen Verhalten widerspiegeln. Wenn ein Cache-Treffer auf einem lokalen Cache erzielt wird, bedeutet dies, daß der Benutzer dieses Web-Dokument bereits mindestens einmal angefordert hat.

  Bei mehrfachem Zugriff auf das gleiche Web-Dokument verringern lokale Caches die Zugriffe auf das Netz und tragen bei einer Anbindung über kostenpflichtige Netzstrecken (Telefonleitung, u.ä.) zur Kostensenkung bei.

  Lokale Caches sind meistens so implementiert, daß ein Teil der Web-Dokumente im Hauptspeicher gehalten wird. Dies sind die Web-Dokumente, die während der laufenden Sitzung als letztes besucht wurden. Hierdurch wird der besonders häufig genutzte Rücksprung zu den zuletzt betrachteten Web-Seiten beschleunigt. Der Hauptspeicher-Cache ist jedoch begrenzt, so daß im Bedarfsfall Web-Dokumente nach dem *first-in-first-out* Prinzip vom Hauptspeicher auf die Festplatte transferiert werden. Der verfügbare Platz im Hauptspeicher und auf der Festplatte kann vom Benutzer (beschränkt durch die Kapazität der vorhandenen Hardware) festgelegt werden.

- **Regionale und nationale Caches**
  Der Verbund von regionalen und nationalen Caches entsteht durch das im vorigen Abschnitt geschilderte Zusammenwirken mehrerer stellvertretender Caches. Sie bilden nach den lokalen Caches die weiteren Stufen in der Caching-Hierarchie.

  Die Effektivität dieser Caches wird durch die lokalen Caches stark eingeschränkt. Cache-Treffer können generell nur aufgrund von zwei Ereignissen erzielt werden. Entweder ein Benutzer fordert ein Web-Dokument an, das er selbst schon einmal

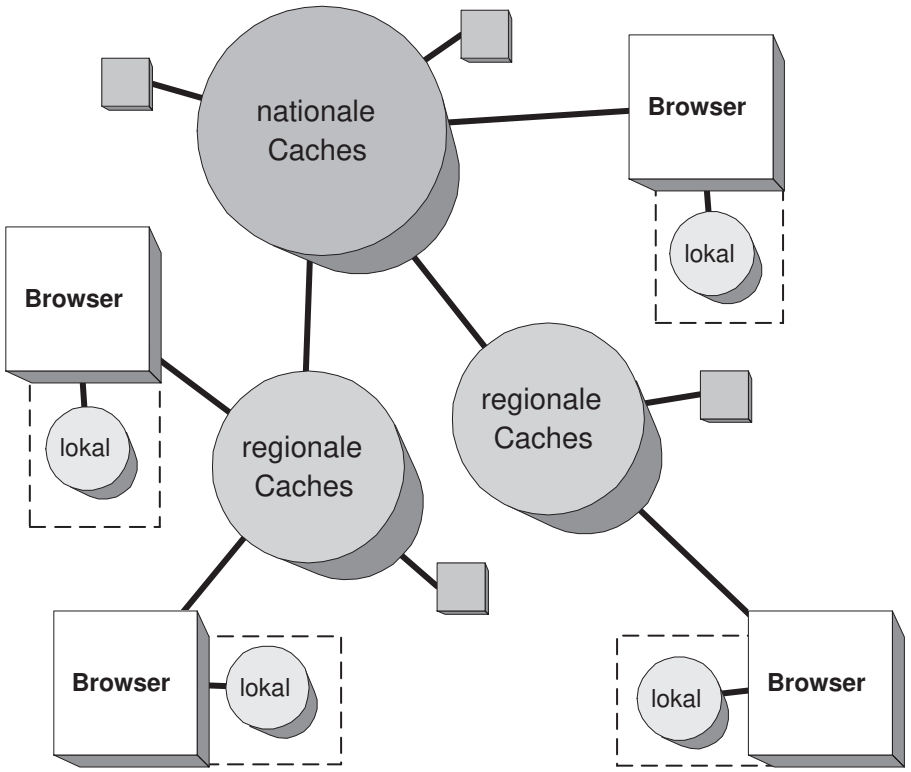

Abbildung 2.4: Verschiedene Ebenen von stellvertretenden Caches

angefordert hat, oder ein Benutzer fordert eine Web-Dokument an, das sich aufgrund der Anforderung eines anderen Benutzer bereits im Cache befindet. Der erste Fall, also die mehrmalige Anforderung durch denselben Benutzer, wird von den lokalen Caches bedient. Nur aufgrund der übrigen Fälle, also der Anforderung eines Web-Dokuments, das sich aufgrund des Zugriffs eines anderen Benutzer im Cache befindet, können Cache-Treffer auf den externen Caches erzielt werden. Um eine zufriedenstellende Anzahl von Cache-Treffern zu erzielen, müssen also mehr Web-Dokumente zwischengespeichert werden, da man nicht mehr von einem homogenen Verhaltensmuster der Benutzer wie bei den lokalen Caches ausgehen kann [ASA+95].

Die Unterscheidung ob ein Cache regional oder national ist sollte nach Größe, Leistung sowie hierarchischer Position im Netz getroffen werden. Eine Trennlinie zwischen regionalen und nationalen Caches ist jedoch nicht eindeutig definiert und beruht eher auf einer intuitiven Zuordnung. Einen Cache, der bei einem regionalen Rechenzentrum läuft, wird man den regionalen Caches zuordnen, während Caches, die eine Hauptverbindung zwischen zwei Ländern darstellen, zu den nationalen Caches gezählt werden. Eine sinnvolle Lastverteilung kann durch die Beschränkung von Caches auf bestimmte Teil-Domänen erzielt werden. Auf einem Rechner, der eine Hauptverbindung in die USA bildet, sollten zum Beispiel hauptsächlich Web-

Dokumente von amerikanischen *Domänen* (.com, .edu, .gov, etc.) zwischengespeichert werden. So speichern die Caches im amerikanischen NLANR-Cacheverbund, die sich an der Ostküste befinden, hauptsächlich Web-Dokumente aus Europa. Die Caches des Verbundes an der Westküste dagegen speichern hauptsächlich Web-Dokumente aus Asien [Cor96].

Durch die Möglichkeit, regionale und nationale Caches in einem Verbund zu organisieren, kann die Last auf mehrere Caches verteilt werden, und bei Ausfall eines Caches bricht nicht der gesamte Caching-Automatismus zusammen. Diese Verbunde sind somit robuster, und die Gesamtkapazität ist besser skalierbar.

2.1.3 Verfahren der Zwischenspeicherung

Der Speicherplatz, der auf einem Cache zur Speicherung von Web-Dokumenten vorhanden ist, ist durch die Größe der verfügbaren Speichermedien beschränkt. Um bei Bedarf freien Speicher zu schaffen, müssen Ersetzungsstrategien implementiert werden, die eine hohe Effektivität des Caches erhalten. Ähnliche Probleme tauchen aber auch in anderen Bereichen der Informatik auf, und man kann auf Verfahren und Ergebnisse aus diesen Bereichen zurückgreifen.
Neben diesem administrativen Problem ist bei der Zwischenspeicherung außerdem zu beachten, daß nicht alle Web-Dokumente, die bei einem Cache ankommen, als *zwischenspeicherbar* klassifiziert werden. Diese *nicht zwischenspeicherbaren* Web-Dokumente müssen ohne Zwischenspeicherung auf dem Cache weitergereicht werden.
Zusätzlich zu den Nutzdaten, fallen bei einer Übertragung eines Web-Dokuments auch noch Meta-Informationen an, die bei einer Auslieferung aus dem Cache rekonstruiert und somit auch zwischengespeichert werden müssen.

2.1.3.1 Ersetzungsstrategien bei Speichermangel

Die Kapazität der Caches, also die Anzahl der Web-Dokumente die zwischengespeichert werden können, ist durch die verwendete Hardware beschränkt. Treten Speicherengpässe auf, müssen vorhandene Web-Dokumente aus dem Cache entfernt werden, um Platz für neue Web-Dokumente zu schaffen. Zu diesem Zweck werden Ersetzungsstrategien eingesetzt, aufgrund derer entschieden wird, welche Web-Dokumente aus dem Cache entfernt werden. Unterschiedliche Ersetzungsstrategien wurden unter anderem im Bereich der Datenbanksysteme ausgiebig untersucht [LS87] und finden auch in Betriebssystemen und in der technischen Informatik Anwendung. Denkbare Kriterien für die Wahl des zu ersetzenden Web-Dokuments sind u.a.

- Zeitpunkt der Speicherung. Hierdurch entsteht zum Beispiel ein *first-in-first-out* Algorithmus der die ältesten Web-Dokumente ungeachtet der erfolgten Zugriffe zuerst aus dem Speicher entfernt.

- Anzahl der Zugriffe. Beispielsweise werden die Web-Dokumente mit den geringsten Zugriffszahlen zuerst aus dem Speicher entfernt, wodurch oft angefragte Web-Dokumente länger im Cache gehalten werden können. Dies kann noch mit einem Zeitfaktor gewichtet werden, um die Effektivität zu steigern.

- Größe der Web-Dokumente. Kleine Web-Dokumente werden zuerst wieder aus dem Cache entfernt, da man davon ausgeht, daß diese bei Bedarf schneller vom Server nachgeladen werden können.

- Eine Kombination mehrerer Kriterien. Die Web-Dokumente werden nach einem Kosten/Nutzen Modell bewertet, wodurch der Wert eines Web-Dokuments ermittelt werden kann [LRV96].

2.1.3.2 Nicht zwischenspeicherbare Web-Dokumente

Nicht alle Web-Dokumente, die einen Cache durchlaufen, dürfen dort zwischengespeichert werden. Sie lassen sich in die Kategorien *zwischenspeicherbar* (*cacheable*) und *nicht zwischenspeicherbar* (*uncacheable*) unterteilen. Genügen die Web-Dokumente, die auf einem Cache auflaufen, einer der folgenden Kriterien, werden sie als nicht zwischenspeicherbar kategorisiert:

- **Dynamische Entstehung.** Web-Dokumente, deren URL auf dynamische Entstehung schließen läßt, sollen von vornherein von der Zwischenspeicherung ausgeschlossen werden. Da eine identische Anforderung ein unterschiedliches Ergebnis liefern kann, ist eine Konsistenz auf dem Cache zwischengespeicherter dynamischer Web-Dokumente nicht gewährleistet. In diese Kategorie fallen insbesondere alle Web-Dokumente, die als Ausgabe eines CGI-Programms [Mor95] erstellt werden.

  Eine, wenn auch recht unzuverlässige Methode zur Erkennung dynamischer Web-Dokumente ist die Analyse des Pfades. Hierbei geht man davon aus, daß sich die CGI-Programme in einem separaten Verzeichnis auf dem Server befinden. Dies ist gängige Praxis, um Mißbrauch zu verhindern. Die Bezeichnung der Verzeichnisse ist jedoch nicht verbindlich geregelt. Der Pfad des angeforderten Web-Dokuments kann also nur auf häufig verwendete Teilzeichenketten wie *cgi-bin*, *win-cgi*, *cgi-shl* o.ä. geprüft werden. Es ist aber nirgends vorgeschrieben, Verzeichnisse in denen sich CGI-Programme befinden unter einem Pfad abzulegen, der einem dieser Muster entspricht. Als Beispiel sei hier nur der Microsoft Internet Information Server genannt, der bei der Installation als Vorgabe das Verzeichnis für die CGI-Programme mit */Scripts/* benennt.

  Verwendet wird dieses Verfahren zum Beispiel vom Harvest Object Cache, welcher keine Web-Dokumente zwischenspeichert, deren Anforderung entweder ein '?' (Beispiel: `GET /cgi-bin/info.cgi?alle`) oder aber die Zeichenkette /cgi-bin/ in der Pfadangabe enthält [BMV96].

- **Explizite Deklaration.** Das Web-Dokument ist explizit als nicht zwischenspeicherbar deklariert worden. Diese Deklaration kann im HTTP-Header erfolgen und wird in Abschnitt 2.1.4 eingehend erläutert.

- **Zugriffsschutz.** Die Daten unterliegen einem Zugriffsschutz. Ist aufgrund des HTTP-Headers ersichtlich, daß der Zugriff auf diese Daten nur aufgrund einer vorher erfolgten Autorisierung genehmigt wurde, dürfen diese Daten in keinem Falle zwischengespeichert werden. Enthält ein Web-Dokument sensible Daten, wäre es fatal, wenn ein Cache es zwischenspeichern und auch ohne Berechtigung zugänglich machen würde.

2.1.3.3 Speicherung von Metainformation

Ein Server schickt zusätzlich Metainformationen zu den Web-Dokumenten mit. Übertragen werden die Metainformationen im HTTP-Header. Das HTTP-Protokoll sieht unter anderem Einträge für ein Verfallsdatum, einen Datentyp und den Zeitpunkt der letzten Änderung vor und kann außerdem Informationen zur Weiterleitung eines Dokumentes und der Verhinderung von Caching enthalten [BLFF95]. Diese Metainformationen müssen bei Zwischenspeicherung des Web-Dokumentes auf dem Cache ebenfalls zwischengespeichert werden, da sie bei einer Auslieferung des Web-Dokumentes aus dem Cache rekonstruiert werden müssen.

2.1.4 Caching-Kontrolle im HTTP-Protokoll

Durch ein Attribut im HTTP-Header ist es möglich Einfluß auf das Caching zu nehmen. Hierbei muß sehr stark zwischen den Versionen HTTP/1.0 [BLFF95] und HTTP/1.1 [FGM+95] des HTTP-Protokolls differenziert werden.
In der HTTP/1.0 Spezifikation ist zu dem genannten Zweck nur der folgende Eintrag definiert:

```
Pragma: no-cache
```

Laut Spezifikation kann dieses Feld im HTTP-Header aber nur von einem Client benutzt werden, um ein Nachladen des angeforderten Web-Dokuments vom Web-Server zu erzwingen. Alle Caches, über die die Anforderung läuft, *sollten*, selbst wenn eine Kopie des angeforderten Web-Dokuments vorhanden ist, die Anforderung an den Server weiterleiten. Die Spezifikation sagt nicht aus, daß Web-Server durch die Verwendung des Pragma-Feldes (s.o.) im HTTP-Header ein Zwischenspeichern des versandten Web-Dokuments auf einem Cache verhindern können. Wichtig ist zu beachten, daß dieser Eintrag lediglich eine Empfehlung darstellt und auch von einem HTTP/1.0 konformen Cache nicht beachtet werden muß.
In der Version 1.1 hat die Unterstützung von Caching an Bedeutung gewonnen. Eine Flußkontrolle für beide Richtungen wird spezifiziert. Clients können erzwingen, daß eine Anforderung an den Server geleitet wird. Server können bei der Beantwortung einer Anforderung das versandte Web-Dokument klassifizieren, wodurch

- Caching generell verhindert (*no-cache*),
- Caching auf lokale Caches beschränkt (*private*) oder
- generell erlaubt (*public*) werden kann.

Neben der erweiterten Funktionalität ist jedoch entscheidend, daß die HTTP/1.1 Spezifikation *vorschreibt*, die Caching-Kontrollmechanismen zu beachten. Dies geht über den Empfehlungscharakter des Caching-Kontrollmechnismus in HTTP/1.0 hinaus. Da die HTTP/1.1 Spezifikation sich noch im Entwurfsstadium befindet, werden HTTP/1.1 konforme Caches, Server und Clients noch nicht verbreitet. Man kann ein Caching aber auch unter HTTP/1.0 verhindern, indem man das Verfallsdatum eines Web-Dokuments auf ein Datum legt, das bereits in der Vergangenheit liegt. Dies war eigentlich gedacht um die Aktualität von Web-Dokumenten zu garantieren (siehe Abschnitt 3.2.1), kann jedoch zu dem oben genannten Zweck mißbraucht werden.

2.2 Caching-Topologien

Regionale/nationale Caches können als Cache-Verbunde organisiert werden. Diese Verbunde können in verschiedenen Topologien angeordnet sein. Zwei der gängigen Topologien werden im folgenden vorgestellt.

2.2.1 Kaskadierte Caches

Kaskadierte Caches sind Serienschaltungen von Caches (siehe Abbildung 2.5). Eine Anforderung wird dabei solange weitergereicht, bis entweder einer der konsultierten Caches eine Kopie der angeforderten Web-Dokument vorliegen hat oder der letzte in der Kette befindliche Cache die Seite vom Server anfordert.

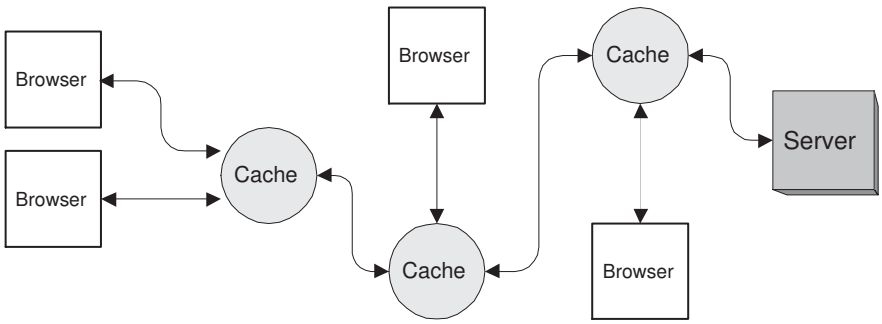

Abbildung 2.5: Kaskadierte Caches

Der Sinn eines kaskadierten Cache-Verbundes ist es die Last auf mehrere Caches zu verteilen und somit die Leistungsfähigkeit im Verbund zu steigern. Wird diese Kette aber zum Beispiel durch den Ausfall eines Caches unterbrochen, kann der Betrieb nur durch Umkonfiguration wieder aufgenommen werden. Deswegen muß diese Topologie als nicht sehr robust eingestuft werden.

2.2.2 Hierarchische Caches

Hierarchische Caches stellen momentan die modernste Topologie dar [CDN+96]. Hier werden mehrere Caches abhängig von ihrer Position und Leistungsfähigkeit hierarchisch angeordnet und untereinander verbunden. Caches, die sich auf derselben Hierarchiestufe befinden, werden *Nachbarn* genannt. Zusätzlich können diese Caches auch einem Cache einer höheren Hierarchistufe, einem *Vater*, zugeordnet (siehe Abbildung 2.6) werden. Trifft eine Anforderung auf einen Cache und ist keine Kopie des angeforderten Web-Dokuments vorhanden, konsultiert dieser seine Nachbarn. Schlägt dies auch fehl, wird die Anforderung an den Vater weitergeleitet. Als Besonderheit ist hier noch zu erwähnen, daß im Falle einer positiven Antwort mehrerer Nachbarn, der Nachbar mit der schnellsten Antwortzeit zu Beantwortung der Anforderung benutzt wird. Weitere Informationen hierzu sind in [Sch95] zu finden.

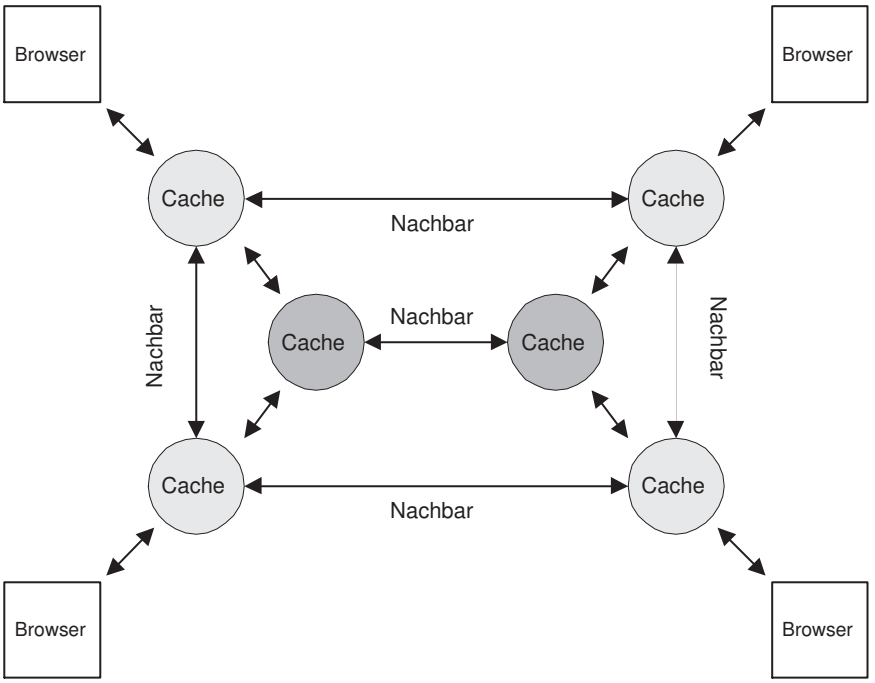

Abbildung 2.6: Hierarchische Caches

Stärken dieser Topologie sind die gute Skalierbarkeit und die Robustheit. Die Last kann gezielt auf mehrere Rechner verteilt werden und der Ausfall eines Caches im Verbund erfordert keine Umkonfiguration wie bei den kaskadierten Caches. Dies wird ermöglicht, indem mehrere unabhängige Caches miteinander kooperieren und gemeinsam Anforderungen bearbeiten [MLB95]. Natürlich wird die Robustheit und Effektivität dieser Topologie durch ein größeres Kommunkationsaufkommen erkauft, welches die Netzlast erhöht.

2.3 Zugriffsprotokollierung

Web-Server protokollieren in der Regel die Anforderungen, die bei ihnen auflaufen und sammeln die Informationen in *Protokolldateien* (*Logfiles*). Dieser Abschnitt befaßt sich mit den vier Standard-Protokolldateien (*access_log, agent_log, error_log, referer_log*) und es wird dargestellt, auf welche Informationen in diesen Protokolldateien zurückgegriffen werden kann. Weiterhin wird auf erweiterte Protokolldateien hingewiesen, die zur Behebung von Schwachpunkten der Standarddateien entwickelt wurden. Die Protokolldateien haben mit der Kommerzialisierung des WWW an Relevanz gewonnen und sind nicht mehr nur als administrative Hilfsmittel zu sehen.

2.3.1 Standard-Protokolldateien und -Formate

Die meistverbreiteten Web-Server speichern die Zugriffe und Ereignisse in vier Protokolldateien ab. Dies sind die Zugriffs-, die Browser-, die Verweis- und die Fehlerprotokolldatei.

2.3.1.1 Server-Zugriffsprotokolldatei

In der Zugriffsprotokolldatei (*access_log*) werden die Zugriffe auf die Web-Dokumente eines Servers protokolliert. Dazu werden die IP-Adresse des Clients, Datum und Uhrzeit, die HTTP-Anforderung, der Status-Code [Con95b] und die Anzahl der übertragenden Bytes protokolliert. Das folgende Beispiel zeigt eine Anforderung der Web-Seite */Idomeneus/Sites/grenoble.html* von einem Rechner mit dem Namen *dbis18.informatik.uni-hamburg.de* am in eckigen Klammern angegebenen Datum. Die Anforderung konnte erfolgreich bedient werden (Status-Code *200*) und es wurden *3421* Bytes übertragen:

```
dbis18.informatik.uni-hamburg.de - - [30/Apr/1996:21:19:49 +0200]
     "GET /Idomeneus/Sites/grenoble.html HTTP/1.0" 200 3421
```

Die Syntax des Beispiels folgt dem *Common Logfile Format* [Con95c], welches von den meisten Web-Servern unterstützt wird. Durch die Verwendung eines einheitlichen Protokollformates ist es möglich, Analyseprogramme ohne Anpassung auf Protokolldateien verschiedener Server anzuwenden.

2.3.1.2 Cache-Zugriffsprotokolldatei

In der Zugriffsprotokolldatei eines Caches (*cache_access_log*) werden Anforderung, die bei diesem Cache eintreffen, protokolliert. Es werden dieselben Daten wie bei einer Server-Zugriffsprotokolldatei gespeichert. Zusätzlich enthalten die Einträge eine Kennung, aus der hervorgeht, ob die Anforderung durch den Cache bedient werden konnte [HSW96]. Hierdurch lassen sich Aussagen über die Trefferquote des Caches machen. Das folgende Beispiel zeigt einen Cache-Treffer und eine Cache-Verfehlung:

```
[30/Apr/1996:21:19:49 +0200] /Idomeneus/Sites/grenoble.html
     dbis18.informatik.uni-hamburg.de 3421 TCP\HIT

[30/Apr/1996:21:22:38 +0200] /Idomeneus/Sites/hamburg.html
     dbis18.informatik.uni-hamburg.de 3421 TCP\MISS
```

2.3.1.3 Browser-Protokolldatei

In der Browser-Protokolldatei (*agent_log*) wird Name und Version des vom Benutzer verwendeten Browsers gespeichert. Die folgende Zeile zeigt an, daß eine Anforderung von einem Browser des Types Netscape Navigator 2.0 (interner Name *Mozilla*) erfolgte. Die Angaben in Klammern geben Aufschluß über das verwendete Betriebssystem. In diesem Fall lief der Browser auf einer X11-Oberfläche auf einem Hewlett Packard Unix-Derivat (*HP-UX*).

```
Mozilla/2.0 (X11; I; HP-UX A.09.05 9000/735)
```

Die Information, welche Browser zum Zugriff auf die Web-Seiten eines Web-Servers verwendet werden, kann bei der Erstellung von Web-Seiten von großer Wichtigkeit sein. Die Autoren können sich auf die HTML-Elemente beschränken, die von den meistgenutzten Browsern unterstützt werden. Diese Einschränkungen sind nötig, da es herstellereigene HTML-Erweiterungen gibt, die nicht von allen Browsern dargestellt werden. Zum Beispiel vermag der Microsoft Internet Explorer erst ab der Version 3.0 *Frames* anzuzeigen. Frames, die das Unterteilen eines Browserfenster gestatten, wurden von Netscape eingeführt und daher anfänglich auch nur von Browsers des Types Netscape Navigator (ab Version 2.0) unterstützt. Stellt sich aufgrund der Einträge in der Browser-Protokolldatei heraus, daß zum Beispiel die meisten Benutzer den Microsoft Internet Explorer in der Version 2 benutzen, so wäre es nicht ratsam bei der Gestaltung der Seiten Frames zu verwenden.

2.3.1.4 Verweisprotokolldatei

Auf Web-Seiten können sich Querverweise (*Links*) auf andere Web-Dokumente befinden. Der Benutzer kann diesen folgen und dadurch das Web-Dokument, auf das verwiesen wird, anfordern. Trifft eine Anforderung bei einem Web-Server ein, die aufgrund eines Querverweises erfolgte, so wird dies in der Verweisprotokolldatei (*referer_log*) festgehalten. Das folgende Beispiel zeigt einen Eintrag, bei dem sich auf der Web-Seite mit der URL *http://guide-p.infoseek.com/Titles.html* ein Querverweis auf das Web-Dokument */ret1/analog/index.html* befindet und von dort angefordert wurde:

```
http://guide-p.infoseek.com/Titles -> /ret1/analog/index.html
```

2.3.1.5 Fehlerprotokolldatei

Die Fehlerprotokolldatei (*error_log*) hält hauptsächlich Ereignisse fest, die sich auf das korrekte Laufen des Servers beziehen, bzw. auf außergewöhnliche Ereignisse schließen lassen. Dies können unter anderem zugreifende Web-Roboter sein, aber auch Links auf Seiten, die ins Leere führen. Sie ist hier nur der Vollständigkeit halber erwähnt, da sie kaum Relevanz für die Caching-Problematik besitzt. Die folgende Zeile ist ein typischen Beispiel, die das korrekte Starten eines Web-Server anzeigt.

```
[Fri Aug 16 15:27:54 1996] httpd: successful restarted
```

2.3.2 Erweiterte Protokolldateien

Die immer stärker verbreiteten Web-Server von Netscape [Cor95a], Microsoft und O'Reilly benutzen alle ein proprietäres Protokolldateiformat, wobei die Inhalte der Protokolldateien konfigurierbar sind. Dadurch kann der Benutzer bestimmen, welche Informationen für ihn relevant sind und deren Protokollierung selektiv einschalten.
In den oben erwähnten Standardprotokolldateien sind die Informationen über eine Anforderung auf die vier Einzelprotokolldateien verteilt. Ein Zusammenführen aller Informationen ist nicht möglich, da der Datumsstempel nur in der Zugriffs- und Fehlerprotokolldatei geführt wird. Die Informationen in der Verweis- und Browser-Protokolldatei können also nicht mit den Einträgen der anderen Protokolldateien verbunden werden. Die erweiterten Protokolldateien umgehen dieses Problem, indem alle eine Anforderung betreffenden Informationen, in derselben Datei gespeichert werden. Die folgende Zeile zeigt ein Beispiel aus einer Protokolldatei eines O'Reilly WebSite Web-Servers. Dieser Eintrag entstand

durch eine Anforderung des Web-Dokuments */neu/jobs.html*, ausgehend von der Web-Seite *http://jan.uni-hamburg.de/neu/navigate.html*. Die Anforderung erfolgte von einem Rechner mit der IP-Adresse *192.42.166.30* und dem aufgelösten Namen *jan.uni-hamburg.de*, wobei ein Netscape Navigator 3.01 (*Mozilla/3.01*) unter Windows 95 verwendet wurde. Der Zugriff erfolgte am *26.01.97* um *22:58:49 Uhr*.

```
01/26/97 22:58:49  192.42.166.30  jan.uni-hamburg.de
     GET /neu/jobs.html  http://jan.uni-hamburg.de/neu/navigate.html
     Mozilla/3.01 (Win95; I)  200
```

Zusätzlich können die bei der Benutzung von Formularen ausgetauschten Daten gespeichert werden, und es ist sogar möglich, den Dialog zwischen Client und Server bis auf die Protokollebene hinab abzulegen. Dies schließt die Verbindungsauf- und Verbindungsabbauphasen sowie die exakte Anzahl der übertragenen Bytes und die Übertragungsdauer mit ein. Man erhält also eine sehr detaillierte Übersicht der Übertragung, was zur Fehlerbehebung sehr nützlich sein kann.

2.3.3 Benutzerdefinierte Protokolldateien

Datenbankanbindungen oder Formularfelder auf Web-Dokumenten ermöglichen das Sammeln von Informationen, die über die Inhalte in den Standard-Protokolldateien hinausgehen. Eine Datenbankanbindung an ein Web-Dokument erfordert in der Regel ein Programm, welches die Daten aus der Datenbank holt und aufbereitet an den Browser zurücksendet. Mit diesem Programm kann gleichzeitig eine Protokolldatei geführt werden, die zum Beispiel die angeforderten Daten oder auch den Zustand der Datenbank enthält. Hierdurch kann man individuelle Protokolldateien erstellen, die auf die Anforderung des Betreibers einer solchen Datenbank zugeschnitten sind. Auch der Inhalt von Formularfeldern wird in keiner der Standard-Protokolldateien berücksichtigt. Bei der Analyse solcher benutzerdefinierter Protokolle kann man auf einen wesentlich größeren Informationsumfang zurückgreifen und somit genauere Ergebnisse erzielen.

2.3.4 Zustandsprotokollierung mit Magic-Cookies

Die bisher vorgestellten Protokolldateien befinden sich alle auf den Servern, bzw. Caches. Eine weitere Möglichkeit besteht darin, auch auf den Clients Informationen abzulegen. Hierzu sendet der Server Daten, *Magic Cookies* oder einfach *Cookies* genannt, an den Browser. Dies kann der Name des Kunden, eine Mitglieds- oder Kreditkartennummer, o.ä. sein. Der Browser speichert das Cookie lokal ab und schickt es bei einer erneuten Anforderung an den Server mit. Hiermit kann man die Nachteile umgehen, die das sonst zustandslose HTTP-Protokoll besitzt. Mit den Cookies kann man den Zustand des Clients aus der Sicht des Servers speichern [Cor95b]. Außerdem kann mit den Cookies die Identifikation eines Benutzers erfolgen und mitgeführt werden. Bei nachfolgend angeforderten Web-Seiten ist diese Information bereits vorhanden und man kann das Angebot individuell auf den angemeldeten Benutzer zuschneiden.
Als Beispiel hierfür seien Katalogsysteme im Internet genannt, bei denen die Bestellungen, die ein Benutzer auf unterschiedlichen Seiten gemacht hat, auf dem Server in einer benutzereigenen Datei gespeichert werden. Dies ist nur möglich, da der Server die Benutzerinformation bei einem Seitenwechsel nicht verliert. Sogar wenn der Benutzer Tage

später dasselbe Katalogsystem wieder besucht, kann auf diese Informationen noch zugegriffen werden. Dies ermöglicht auch das Erstellen von Benutzerprofilen.
Ähnlich wie Web-Dokumente können auch Cookies mit einem Verfallsdatum versehen werden, damit diese, wenn die Informationen nicht mehr benötigt werden, nicht unnötigerweise bei jeder Transaktion wieder mitgeführt werden.

Kapitel 3

Auswirkungen von Caching-Verfahren

Das Caching im Internet bringt neben den Vorteilen der Geschwindigkeitssteigerung, auch Probleme mit sich. Insbesondere die Verfälschung der Zugriffszahlen auf Web-Dokumente in den Protokolldateien hat zur Folge, daß Statistiken, die auf der Grundlage dieser Protokolldateien erstellt wurden, an Aussagekraft verlieren [Gol95]. Aber auch Probleme der Aktualität von angeforderten Web-Dokumenten sind Folgen der Caching-Technologie.
Hinter dem ersten Problem verbirgt sich ein kommerzielles und hinter den weiteren ein allgemeines Interesse. Dieses Kapitel beschreibt die Probleme, und es werden Lösungen in Ansätzen aufgezeigt, die Abhilfe schaffen können.

3.1 Verfälschung der Zugriffszahlen auf Web-Dokumente

Anforderungen für ein Web-Dokument werden von Caches abgefangen und von diesen bedient. Da diese Anforderungen die Server nicht erreichen, können sie auf diesen auch nicht protokolliert werden. Die in den Server-Protokolldateien vermerkten Zugriffe auf ein Web-Dokument entstehen also ausschließlich durch direkte Zugriffe auf dieses Web-Dokument und weisen daher nicht die gesamte Anzahl der Zugriffe aus. Dies stellt kein Problem dar, solange nur die auf diesen Server erfolgten Zugriffe interessieren, um zum Beispiel Aussagen über die Auslastung des Servers machen zu können.
Kommerzielle Anbieter haben hier jedoch ein ganz anderes Interesse. Für sie ist die Anzahl der Gesamtzugriffe entscheidend, da der Wert einer Web-Seite als Werbeträger unmittelbar von dieser abhängt. Dies ist vergleichbar mit Werbeflächen innerhalb von Zeitschriften/Zeitungen, deren Wert auch von der Auflage bzw. der Verbreitung abhängt. Hier gibt es ein vergleichbares Problem, da verkaufte Exemplare durchaus von mehreren Personen gelesen werden. Beispielsweise können Zeitschriften, die im Wartezimmer eines Arztes ausliegen, höher bewertet werden als Zeitschriften, die in private Haushalte geliefert werden. Da die Herstellung und der Vertrieb zumeist in einer Hand liegen, stehen die zur Korrektur benötigten Informationen aber eher zur Verfügung.
Im Internet ist nicht vorhersehbar, welche Verbreitung ein Web-Dokument erfährt, da der Rechner, der ein Web-Dokument anfordert ein Einzelplatzrechner, ein Cache oder eventuell ein Verbindungsrechner sein kann, der ein ganzes Subnetz hinter sich verbirgt (siehe Kapitel 3.3).

Der gesamten Problematik der Korrektur von Zugriffszahlen liegt ein kommerzielles Interesse zugrunde und Lösungsverfahren werden hauptsächlich von den kommerziellen Anbietern im Internet vorangetrieben.
In den folgenden Abschnitten sollen Lösungsmöglichkeiten für die beschriebene Problematik in Ansätzen aufgezeigt werden. Ziel aller Verfahren ist es, die in den Protokolldateien angegebene Anzahl von Zugriffen auf ein Web-Dokument um die Anzahl der durch Caches abgefangenen Zugriffe zu korrigieren.

3.1.1 Modellbildung

Bei einer Modellbildung wird versucht, das Verhalten der involvierten Caches zu analysieren, um eine Aussage über den Einfluß dieser Cache-Topologie auf die Zugriffe auf einen Web-Server zu machen. Für einen Web-Server muß also bekannt sein, welche Caches auf die Zugriffe auf diesen Server Einfluß nehmen und welche Trefferraten auf den jeweiligen Caches erzielt werden. Außerdem muß für jeden dieser Caches bekannt sein, von wie vielen Browsern er verwendet wird. Besonders wenn der Cache gleichzeitig die Aufgabe eines Firewalls erfüllt, wäre diese Aufgabe nur schwer zu bewältigen, da es in der Regel nicht gewünscht ist, Art und Anzahl der hinter dem Firewall befindlichen Rechner bekanntzugeben. IP-Maskierung (siehe Kapitel 3.3) ist ein weiteres Problem für die Modellbildung, da hiermit Subnetze verborgen werden können und man somit auf die Auskunft des Betreibers des Subnetzes angewiesen ist. Dieser müßte die Struktur oder zumindest die Anzahl der angeschlossenen Rechner des internen Subnetzes bekanntgeben, was durchaus Sicherheitsrisiken in sich bergen kann. Auch hier muß man sich wieder die Interessenlage vor Augen führen, die nicht beim Betreiber des Subnetzes sondern nur beim Betreiber des Servers liegt. Gleiches gilt für die Betreiber eines Caches in einem Cache-Verbund, da auch hier die Information über Art und Struktur des Caches nur den Betreibern bekannt ist, wodurch diese Information einen Wert für den Betreiber des Web-Servers darstellt. Es ist also gut möglich, daß diese Informationen eines Tages gegen Bezahlung weitergegeben werden. Ein Indiz hierfür ist das sehr zurückhaltende Verhalten der Betreiber von Cache-Verbunden, die Spezifikation ihres Verbundes offenzulegen. Diese Schwierigkeiten und nicht zuletzt die Probleme der Verifikation eines aufgestellten Modells, lassen diesen Ansatz als nicht praktikabel erscheinen.

3.1.2 Benachrichtigung der Server durch die Caches

Die Gesamtzahl der Zugriffe auf ein Web-Dokument entspricht der Summe der Zugriffe auf das Original auf dem Web-Server und der Zugriffe auf die Kopien des Web-Dokuments auf den Caches. Wenn man die Protokolleinträge der Caches den Servern zugänglich macht, kann man mit dieser Zusatzinformation die Einträge in den Server-Protokolldateien korrigieren. Die Übertragung dieser Daten führt aber zu einer Mehrbelastung des Netzes und ist somit nicht zu empfehlen. Außerdem haben die lokalen Caches die höchste Effektivität im Cache-Verbund, wodurch es nötig wäre, auch diese Informationen an den Web-Server zu senden.
Da zumindest die erste Teilstrecke vom Client zum Server durch den Benutzer zu bezahlen ist, wird diese unnötige Netzbelastung nur schwer zu rechtfertigen sein. Man muß sich immer vor Augen führen, daß der Nutzen einer Korrektur der Zugriffszahlen allein beim Betreiber des Web-Servers zu sehen ist. Alle Korrekturverfahren, bei denen eine Korrektur

zusätzliche Netzbelastung erzeugt, müssen deshalb kritisch betrachtet werden, wenn auch Kosten beim Benutzer entstehen.

3.1.3 Verwendung nicht zwischenspeicherbarer Web-Dokumente

Wie in Abschnitt 2.1.3.2 erläutert, gibt es zwischenspeicherbare und nicht zwischenspeicherbare Web-Dokumente. Nicht zwischenspeicherbare Web-Dokumente müssen bei jeder Anforderung vom Web-Server geladen werden und können nicht auf Caches zwischengespeichert werden. Dadurch wird die Protokollierung der Zugriffe auf diese Web-Dokumente nicht verfälscht. Wenn nun eine eindeutige Zuordnung eines solchen nicht zwischenspeicherbaren Web-Dokuments zu einer Web-Seite möglich ist, kann hiermit die Zugriffsprotokollierung dieser Web-Seite korrigiert werden. Diese Technik wirkt der Idee, die hinter dem Caching steht, natürlich entgegen, da es Ziel des Cachings ist, Zugriffe auf die Web-Server zu verringern. Die Untersuchung im nächsten Kapitel beschäftigt sich ausführlich mit den Vor- und Nachteilen dieser Technik und stellt ein konkretes Verfahren auf dieser Basis vor.

3.2 Versionsabgleich von Web-Seiten

Web-Dokumente können sich ändern, und es kann somit passieren, daß eine Version eines Web-Dokuments auf einem Cache zwischengespeichert ist, welche nicht mehr aktuell ist und somit auch nicht mehr ausgeliefert werden sollte. Im HTTP-Protokoll sind drei Möglichkeiten vorgesehen, um diesem Aktualitätsproblem entgegenzuwirken. Ein Verfallsdatum kann angegeben werden, der Zeitpunkt der letzten Änderung kann mitgegeben werden, oder bei einer Anforderung kann die Auslieferung eines Web-Dokuments davon abhängig gemacht werden, ob es nach einem angegebenen Datum verändert worden ist.

3.2.1 Angabe eines Verfallsdatum

Mit Hilfe des *Expires-Feldes* (Beispiel: `Expires: Fri, 29 Nov 1996 00:00:00 GMT`) im HTTP-Header kann ein Verfallsdatum für das Web-Dokument angegeben werden. Der Cache speichert dieses Datum zusammen mit dem Web-Dokument ab. Erreicht eine Anforderung für dieses Web-Dokument den Cache zu einem Zeitpunkt, der nach dem Verfallsdatum liegt, so wird das Web-Dokument automatisch mit einer bedingten Anforderung (siehe Abschnitt 3.2.3) vom originären Web-Server geholt und die lokale Kopie aktualisiert. Das Expires-Feld wird jedoch selten genutzt, da eine Verfallsdatum für Web-Dokumente nur in Ausnahmefällen, wie zum Beispiel bei den Seiten einer Online-Zeitung, die regelmäßig erscheint, gesichert angegeben werden kann.
Web-Dokumente, deren Verfallsdatum erreicht ist, können automatisch aus dem Cache entfernt werden, um so wieder freien Speicher für neue Web-Dokumente zu schaffen.
Um eine weitere Leistungssteigerung zu erzielen, können abgelaufene Web-Dokumente, auch ohne daß eine Anforderung vorliegt, vom Cache aktualisiert werden (*cache-prefetching*). Hierdurch kann der Inhalt des Caches auf einem aktuellen Stand gehalten werden. Die Gefahr hierbei ist, daß Web-Dokumente nachgeladen werden, die vielleicht nie mehr angefordert werden, was eine überflüssige Netzbelastung erzeugt. Es sollte also von der Frequenz der Anforderung für ein Web-Dokument abhängig gemacht werden, ob dieses nach Ablauf automatisch nachgeladen wird oder nicht.

Der Vorteil dieses Verfahrens ist die geringere Wartezeit für die Benutzer, da das Nachladen eines abgelaufenen Web-Dokuments nicht erst erfolgt, wenn es erneut angefordert wird. Außerdem kann dieses Nachladen so koordiniert werden, daß Zeiten geringer Netzbelastung genutzt werden, wodurch eine bessere Ausnutzung der vorhandenen Ressourcen und Kapazitäten möglich wird.

3.2.2 Angabe des Datums der letzten Änderung

Das *Last-Modified-Feld* im HTTP-Header erlaubt die Angabe des letzten Änderungsdatums eines Web-Dokuments. Das Änderungsdatum wird verwendet, um bei Dokumenten, die nicht mit einem Verfallsdatum ausgestattet sind oder ausgestattet werden können, dynamisch ein Verfallsdatum abzuschätzen. Hierbei geht man von der Annahme aus, daß kürzlich geänderte Web-Dokumente mit einer größeren Wahrscheinlichkeit bald wieder geändert werden, als solche, die seit langem nicht mehr geändert wurden.

3.2.3 Überprüfung der Aktualität eines Web-Dokuments

Bei einer Anforderung kann im *If-Modified-Since-Feld* des HTTP-Headers ein Datum angegeben werden, wodurch die Anforderung zu einer *bedingten Anforderung* wird. Ein Server liefert das angeforderte Web-Dokument dann nur zurück, wenn es seit dem angegebenen Datum geändert worden ist. Ist dies nicht der Fall, wird nur eine Meldung zurückgeschickt die dem Cache mitteilt, daß das Web-Dokument seit dem angegebenen Datum nicht geändert wurde (Rückgabewert 304, *not modified*)[Con95b]. Diese beiden möglichen Reaktionen eines Servers werden in Abbildung 3.1 verdeutlicht.

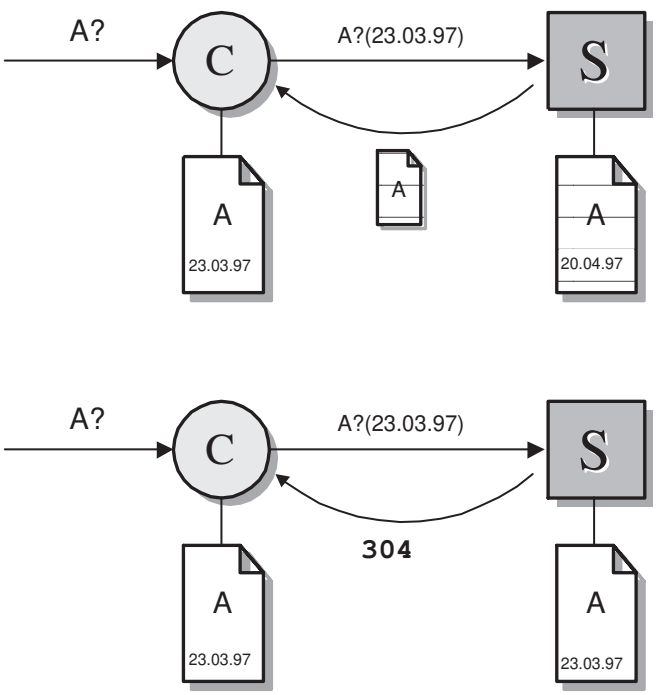

Abbildung 3.1: Verhalten eines Web-Servers bei einer bedingten Anforderung

Erhält ein Cache also aufgrund einer bedingten Anforderung das angeforderte Web-Dokument, so kann die lokale Kopie als abgelaufen angesehen und durch die neue ersetzt werden. Empfängt der Cache hingegen die Meldung, daß das Web-Dokument nicht geändert worden ist, so kann die lokale Kopie als aktuell betrachtet und weiterhin zur Beantwortung von Anforderungen benutzt werden.
Caches sind in der Regel so konfigurierbar, daß sie entweder nur dann bedingte Anforderungen verwenden, wenn die lokale Kopie eines Web-Dokuments abgelaufen ist (siehe Abschnitt 3.2.1), oder jedesmal wenn eine Anforderung aufläuft [BMV96].
Interessant ist die letztere Variante im Sinne der Zugriffsprotokollierung, weil die Anforderungen für ein Web-Dokument immer zum Web-Server gelangen und hier protokolliert werden. Die normalerweise durch Caching entstehende Verfälschung der Zugriffszahlen auf ein Web-Dokument in den Server-Protokolldateien entfällt hierdurch. Es entsteht aber eine zusätzliche Belastung des Netzes, da selbst in dem Falle eines Cache-Treffers, eine Verbindung zwischen dem Web-Server und dem Cache aufgebaut werden muß. Die Aktualität des versendeten Web-Dokumentes kann dadurch aber garantiert werden.

3.3 Verwendung von IP-Maskierung

Alle Rechner, die am Internet angeschlossen sind, benötigen eine Adresse, die eine weltweit eindeutige Referenzierung erlaubt. Diese *IP-Adresse* (Internet-Protokoll-Adresse), besteht aus 4 Bytes, wodurch die theoretische Anzahl der möglichen IP-Adressen auf 256^4 beschränkt ist. Um nicht unnötig IP-Adressen zu verschwenden ist es deshalb möglich Teilnetze über einen einzelnen Verbindungsrechner am Internet anzuschließen, wie dies auch durch Firewalls erfolgt. Dieser Verbindungsrechner ersetzt nun die IP-Adresse des Absender in allen Datenpakete, die aus dem Subnetz kommen, durch die eigene und leitet diese modifizierten Pakete ins Internet. Dadurch tritt nur der Verbindungsrechner im Internet in Erscheinung, weswegen auch nur dieser eine der offiziellen IP-Adressen benötigt, während die Rechner innerhalb des Subnetzes beliebige IP-Adressen haben können. Dieses Verfahren bezeichnet man als *IP-Maskierung* (*IP-Masquerading*). Abbildung 3.2 zeigt die Anforderungen von drei unterschiedlichen Rechner mit den IP-Nummern `192.42.166.10/11/12` die auf dem Verbindungsrechner auflaufen. Dieser ersetzt die IP-Nummern mit der eigenen, offiziellen IP-Nummer (`139.62.100.20`) und leitet die Anforderungen an den Server weiter. Hier ist nun nicht mehr ersichtlich, daß diese Anforderungen von drei unterschiedlichen Rechnern, respektive unterschiedlichen Benutzern, stammen.
Der Vorteil ist, daß mit diesem Verfahren die Zahl der an das Internet anschließbaren Rechner erheblich gesteigert wird. Weiterhin wird hierdurch die Existenz des Subnetzes für Rechner im Internet verborgen, was zur Sicherheit des Subnetzes beiträgt.
Für die Zugriffsprotokollierung bringt dies neue Probleme, da man aufgrund der IP-Adresse nicht mehr eindeutig auf einen Rechner schließen kann. Wurde zum Beispiel laut Zugriffsprotokolldatei ein Web-Dokument n-mal angefordert, so kann man keine Aussage machen, ob diese Zugriffe n mal von demselben Rechner erfolgt sind oder ob n Rechner die Seite je einmal angefordert haben.

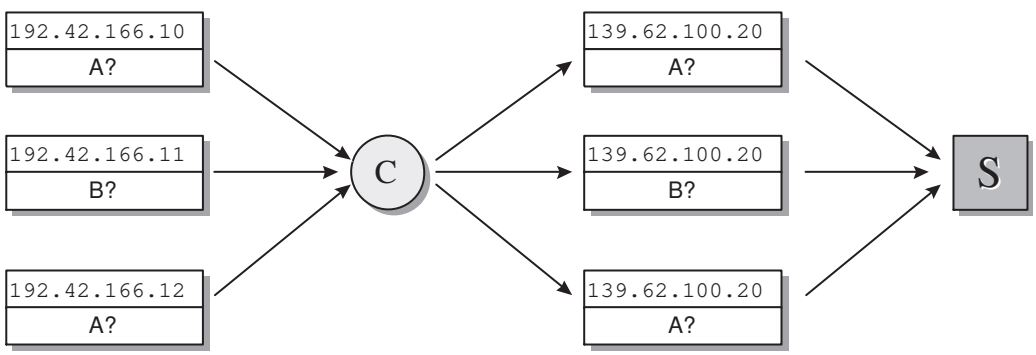

Abbildung 3.2: IP-Maskierung durch einen Cache

Kapitel 4

Korrektur von Zugriffszahlen durch nicht zwischenspeicherbare Web-Dokumente

Im letzten Kapitel wurde der Einfluß des Cachings auf die Zugriffsprotokollierung beschrieben. Dieses Kapitel befaßt sich mit einem Ansatz zur Korrektur der durch das Caching entstehenden Verfälschungen. Mit Hilfe von nicht zwischenspeicherbaren Web-Dokumenten (siehe Abschnitt 2.1.1) wird der Caching-Mechanismus umgangen und so eine Korrektur der aus den Protokolldateien gewonnenen Ergebnisse ermöglicht. Die Idee und Umsetzung eines entsprechenden Verfahrens wird in diesem Kapitel erklärt, Stärken und Schwachpunkte diskutiert und ein Einsatz auf zwei Versuchs-Servern demonstriert. Weiterhin wird eine Modifikation zur Verbesserung des Verfahrens auf der Basis der Ergebnisse dieses Kapitels vorgeschlagen.

4.1 Idee des Verfahrens

Wie in Abschnitt 2.1.1 erläutert, werden Web-Dokumente, die als *nicht zwischenspeicherbar* klassifiziert sind, von Caches nicht zwischengespeichert. Dadurch gelangen alle Anforderungen zum Server und die Zugriffszahlen auf diese Web-Dokumente werden nicht verfälscht. Dies kann man sich zunutze machen, um die Zugriffszahlen von zwischenspeicherbaren Web-Dokumenten zu korrigieren. Hierzu bettet man ein nicht zwischenspeicherbares Web-Dokument, in diesem Fall ein dynamisch erzeugtes Bild, in eine Web-Seite ein. Abbildung 4.1 zeigt den Ablauf einer Anforderung für diese modifizierte Web-Seite. Zuerst wird die Seiteninformation im Web-Dokument A angefordert und durch den Server (S) ausgeliefert. Die Seiteninformation wird durch den Browser ausgewertet und das eingebettete Bild F wird nachgeladen. Danach setzt der Browser die Web-Seite zusammen und stellt diese dar. Bei einem erneuten Zugriff auf diese Web-Seite wird die Anforderung von A vom Cache (C) bedient, da sich seit der letzten Transaktion eine Kopie von A im Speicher befindet. Die nachfolgende Anforderung von F wird aber vom Cache an den Server weitergereicht, da F als nicht zwischenspeicherbar klassifiziert und somit auf dem Cache nicht zwischengespeichert wurde. Die Zugriffszahlen des Bildes werden daher nicht durch Caching verfälscht und spiegeln die Gesamtzugriffe auf die Web-Seite wider.

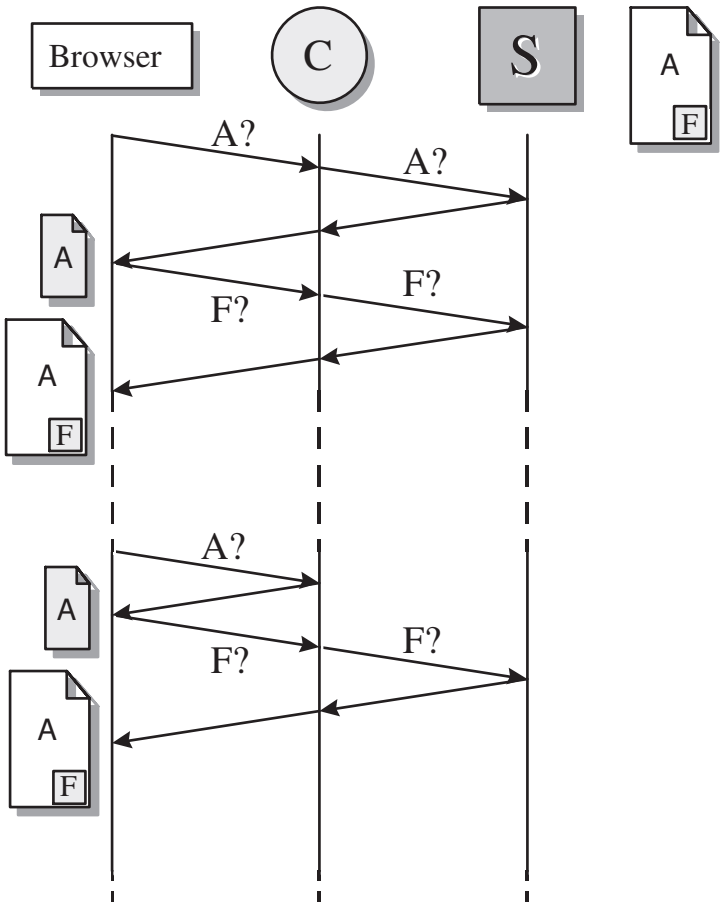

Abbildung 4.1: Verlauf der Anforderung einer Web-Seite mit eingebettetem nicht zwischenspeicherbaren Web-Dokument

4.2 Umsetzung des Verfahrens

Wichtige Forderungen für die Akzeptanz des Verfahrens sind, daß es

- den Normalbetrieb möglichst wenig beeinträchtigt,
- keine Auswirkungen auf das Erscheinungsbild der Web-Seite hat und
- die Effizienz des Web-Servers nicht maßgeblich verringert.

Damit das Verfahren universell einsetzbar ist, sollte das Erscheinungsbild einer Web-Seite hierdurch nicht beeinflußt werden. Das auf der Web-Seite eingebettete Bild sollte also unsichtbar sein, und die Größe der zu übertragenen Daten muß möglichst gering ausfallen. Daher wird ein 2x2 Pixel großes, transparentes Bild verwendet. Das Bild wird im GIF89a-Format kodiert und ist in diesem Format mit 43 Bytes das kleinste mögliche Bild [Bor95]. Solche Bilder werden auch als *Fliegen* bezeichnet. Um das Caching gezielt zu umgehen, muß dem Bild im HTTP-Header noch ein Verfallsdatum mitgegeben werden, welches vor dem Datum der Erzeugung liegt. Wie in Kapitel 2.1.4 bereits geschildert, ist dies die einzige

Möglichkeit mit HTTP/1.0 ein Zwischenspeichern eines Web-Dokuments auf einem Cache zu verhindern. Mit Durchsetzung von HTTP/1.1 ergeben sich elegantere Möglichkeiten. Die folgenden Zeilen geben den C-Quelltext des verwendeten Programms an:

`fly.exe :`

```
1: #include <stdlib.h>
2: #include <stdio.h>
3:
4: #define size 43
5:
6: # Kodierung des 2x2 Pixel großen, transparenten Bildes im GIF89a Format
7: const char gif[]=
8: { \
9: 0x47,0x49,0x46,0x38,0x39,0x61,0x02,0x00, \
10: 0x02,0x00,0x80,0x00,0x00,0xff,0xff,0xff, \
11: 0x00,0x00,0x00,0x21,0xf9,0x04,0x05,0x00, \
12: 0x00,0x00,0x00,0x2c,0x00,0x00,0x00,0x00, \
13: 0x02,0x00,0x02,0x00,0x00,0x02,0x02,0x84, \
14: 0x51,0x00,0x3b};
15:
16:
17: int main()
18: {
19: int i;
20:
21: # Angabe eines Verfallsdatums für das Bild
22: printf ("Expires: Fri, 29 Nov 1996 00:00:00 GMT\n");
23:
24: # Deklaration des gesendeten Daten als Bild im GIF-Format
25: printf ("Content-type: image/gif \n\n"); % 68 Zeichen
26:
27: # Übertragen der 43 Bytes Bilddaten
28: for (i=0; i<size; i++) printf("%c" ,gif[i]);
29:
30: return 0;
31: }
```

Um das Bild auf der Web-Seite einzubetten, wird ein HTML-Befehl verwendet, das *Image-Tag*. Hier kann die Quelle eines Bildes angeben werden, wobei entweder die URL eines statischen Bildes oder die URL eines CGI-Programms, welches das Bild erzeugt, angegeben wird. In diesem Fall wird die URL des oben beschriebenen Programms `fly.exe` angeben, welches die Fliege erzeugt. Wichtig ist, daß nur durch die Verwendung eines Programms ein explizites Verfallsdatum übergeben werden kann (Programmzeile 22) und es daher nicht genügt ein statisches Bild anzugeben. Das folgende Beispiel zeigt die Seitenbeschreibung einer Web-Seite mit eingebetteter Fliege:

```
<HTML>
<HEAD>
<TITLE>Demonstrations-Homepage</TITLE>
</HEAD>
<BODY>
<H1>Dies ist eine Homepage</H1>
Guten Tag auf der Homepage.
<IMG SRC="/cgi-bin/fly.exe">
</BODY>
</HTML>
```

Abbildung 4.2 zeigt die Darstellung dieser Web-Seite in einem Browser, wobei die Fliege wie gewünscht nicht zu sehen ist.

Abbildung 4.2: Darstellung der Demonstrations-Homepage mit eingebetteter Fliege

4.3 Probleme des Verfahrens

Das vorgestellte Verfahren hat einige Schwachpunkte, wodurch die Zugriffszahlen auch nach den mit Hilfe der Fliege vorgenommenen Korrekturen noch fehlerbehaftet sind. Das Problem des *Image-Off Faktors* bewirkt, daß Zugriffe auf eine Web-Seite nicht protokolliert werden, obwohl sie mit einer Fliege versehen sind. Weiterhin gibt es Probleme bei der Benutzung von *Frames*. Außerdem führt dieses Verfahren natürlich zu einer Mehrbelastung des Internets, da das Caching gezielt umgangen wird.

4.3.1 Der Image-Off Faktor

Wie bereits erläutert, werden die Fliegen als Bilder auf den zu untersuchenden Web-Seiten eingebettet. Diese Bilder werden in einer separaten Transaktion vom Server angefordert (siehe Abbildung 4.1). Wenn dies automatisch durch den Browser erfolgt, kann man die

Zugriffe auf die Fliege mit den realen Zugriffen auf die Web-Seite gleichsetzen. Um Übertragungszeiten zu sparen, kann man dieses automatische Nachladen bei den Browsern allerdings abschalten. Der Benutzer kann die Bilder in diesem Fall selektiv nachladen. Wenn der Benutzer also eine Web-Seite anfordert und diese von einem Cache erhält, muß er manuell die Fliege anfordern, um den Zugriff auf die Web-Seite für den Web-Server sichtbar zu machen.
Abbildung 4.3 zeigt die Demonstrations-Homepage aus Kapitel 4.2, wobei diesmal das automatische Nachladen von eingebetteten Elementen im Browser abgeschaltet wurde. Nun wird ein Platzhalter für die Fliege angezeigt, um dem Benutzer mitzuteilen, daß sich hier ein Bild befindet, welches nachgeladen werden kann. Unterbleibt dies, kann die Zugriffsprotokollierung auf zwei Arten verfälscht werden.

1. Wenn sich die angeforderte Web-Seite in einem Cache befindet und die Anforderung von diesem Cache bedient wird, wird weder für die Web-Seite noch für die Fliege ein Zugriff auf dem Web-Server registriert. Dies hat zur Folge das einige Zugriffe auf die Web-Seite nach wie vor nicht registriert werden.

2. Wenn die Web-Seite direkt von Server angefordert wird, wird hier ein Zugriff auf die Web-Seite, aber keiner auf die Fliege registriert. Geschieht dies häufiger, können unter Umständen mehr Zugriffe auf die Web-Seite, als auf die darauf befindliche Fliege protokolliert werden.

Dieses Problem, also die Verfälschung der Zugriffszahlen bei Abschaltung des Nachladens von Bildern, wird als *Image-Off Faktor* bezeichnet. Die Frage, ob man den Image-Off Faktor bestimmen kann, ist noch offen und soll auch nicht Inhalt dieser Arbeit sein. Außerdem hat dies noch einen unschönen Nebeneffekt zur Folge, da der Platzhalter für ein eingebettetes Bild das Erscheinungsbild einer Web-Seite sehr stören kann (siehe Abbildung 4.3).

Abbildung 4.3: Demonstrations-Homepage ohne automatisches Nachladen

4.3.2 Probleme bei der Benutzung von Frames

Netscape führte mit der Version 2.0 ihres Browser eine HTML-Erweiterung ein, mit deren Hilfe man das Browser-Fenster unterteilen und separate Web-Seiten in den einzelnen Teilfenstern darstellen kann. Diese Teilfenster werden als *Frames* bezeichnet. Die folgenden Zeilen zeigen ein einfaches Beispiel, bei dem das Fenster der Browser in zwei Spalten geteilt wird, wobei zwei unterschiedliche Web-Seiten (`links.html` und `rechts.html`) in den Spalten dargestellt werden.

`frame.html:`

```
<html>
  <frameset cols="*,*" rows="*">
    <frame scrolling=no src="links.html">
    <frame scrolling=no src="rechts.html">
  </frameset>
</html>
```

`links.html:`

```
<html>
  <body>
    <img src="/cgi-bin/fly.exe">
  </body>
</html>
```

`rechts.html:`

```
<html>
  <body>
    <img src="/cgi-bin/fly.exe">
  </body>
</html>
```

Bei einem Zugriff auf `frame.html` werden die Web-Seiten `links.html` und `rechts.html` angezeigt. Beide Seiten enthalten eine Fliege. Man sollte erwarten, daß in den Protokolldateien des Web-Server auch zwei Zugriffe auf `/cgi-bin/fly.exe` registriert werden.
Dieses Beispiel wurde mit dem Netscape Navigator 3 und dem Microsoft Internet Explorer 3.0 getestet, und es wurde bei beiden Browsern nur ein einzelner Zugriff auf `/cgi-bin/fly.exe` auf dem Web-Server registriert. Der Browser geht vermutlich davon aus, daß ein dynamisch erzeugtes Bild, das fast zeitgleich zweimal angefordert werden soll, ein identisches Ergebnis bei beiden Anforderung liefern würde und es somit genügt, das Bild einmal anzufordern. Um dieses Verhalten zu umgehen, muß man das Programm (*fly.exe*), das die

Fliege erzeugt, entweder für jede Seite umbenennen, oder bei der Anforderung einen Parameter übergeben, der den Unterschied in den Aufrufen für den Browser ersichtlich macht. Ersetzt man die beiden Zeilen einmal durch `<img src="/cgi-bin/fly.exe L">` und auf der anderen Web-Seite durch `<img src="/cgi-bin/fly.exe R">`, so kann man bei einem Aufruf von `frame.html` tatsächlich zwei Zugriffe auf `/cgi-bin/fly.exe` verzeichnen. Dieses Problem ist also keineswegs kritisch, man muß sich bei der Verwendung von Frames dessen nur bewußt sein und das Problem durch geeignete Maßnahmen, wie z.B. der Parametrisierung der Aufrufe, umgehen.
Dieses Problem ist keine größere Schwachstelle des Verfahrens, da es, wie erläutert wurde, sehr leicht zu umgehen ist. Es zeigt aber, wie wichtig es für das korrekte Funktionieren des Verfahrens ist, daß sich alle beteiligten Komponenten (Browser, Caches, Server) so verhalten, wie es bei dem Verfahren implizit angenommen wird.

4.3.3 Unnötiges Datenaufkommen

Wenn eine Fliege vom Web-Server angefordert wird, gibt dieser zusammen mit den Bilddaten noch den folgenden HTTP-Header zurück, in dem unter anderem auch das für die Caching-Umgehung notwendige Verfallsdatum zurückgegeben wird.

```
HTTP/1.0 200 OK
Date: Sunday, 16-Feb-97 14:22:52 GMT
Server: WebSite/1.1e
Allow-ranges: bytes
Expires: Fri, 29 Nov 1996 00:00:00 GMT
Content-type: image/gif
Content-length: 43
<43 Bytes Bilddaten>
```

Dieser HTTP-Header ergibt zusammen mit den Bilddaten ein zu übertragendes Gesamtvolumen von 255 Bytes. Diese Daten müssen bei jeder Anforderung einer Web-Seite, auf der sich eine Fliege befindet, übertragen werden und zwar über die gesamte Netzstrecke zwischen dem Web-Server und dem Browser. Hinzu kommen außerdem noch die für den Verbindungsauf- und -abbau benötigten Ressourcen [Ste94].

4.4 Exemplarische Messungen

In diesem Abschnitt werden die Messungen einer Anwendung von Fliegen auf zwei Web-Servern vorgestellt. Eingangs werden die besonderen Charakteristika der untersuchten Web-Server beschrieben und danach die bereits beschriebenen Probleme des Verfahrens anhand der ermittelten Daten verdeutlicht.

4.4.1 Charakteristika der untersuchten Web-Server und Web-Seiten

Als Versuchs-Plattformen standen zwei Web-Server zur Verfügung. Der erste ist der Web-Server des Arbeitsbereiches DBIS am Fachbereich Informatik der Universität Hamburg (im weiteren DBIS-Server genannt). Die Daten, die auf diesem Web-Server abrufbar sind, richten sich hauptsächlich an Studenten des Arbeits- bzw. Fachbereichs, sowie an Personen, die an den Publikationen und Aktivitäten des Arbeitsbereiches interessiert sind.

Zugriffe der DBIS-Mitglieder finden in der Regel lediglich unter Benutzung der internen Caches statt, da sich die benutzten Rechner im selben Subnetz befinden. Außerdem werden die internen Caches der Mitglieder des Arbeitsbereiches DBIS jede Nacht automatisch gelöscht.

Diese Eigenarten der Umgebung des DBIS-Servers sind nicht typisch für das WWW, wodurch es interessant ist zu beobachten, welche Ergebnisse die Anwendung von Fliegen in einer atypischen Umgebung liefert, da sie ja als universelles Mittel zu Korrektur von Zugriffszahlen einsetzbar sein sollte.

Der zweite untersuchte Server ist der Web-Server einer Hamburger Tageszeitung (im weiteren Zeitungs-Server genannt). Die Web-Seiten dieses Server sind durchgängig öffentlich zugänglich, die Benutzergruppe ist also nicht eingeschränkt. Im Vergleich mit dem DBIS-Server haben wir es hier generell mit einem größeren, inhomogeneren Benutzerkreis zu tun. Da es sich um das Online-Angebot einer Tageszeitung handelt, werden die Web-Seiten sechsmal in der Woche erneuert. Auf die beteiligten Caches wird, im Gegensatz zum DBIS-Server, keinerlei Einfluß genommen.

4.4.2 Messungen der ersten Versuchsanordnung

In der ersten Versuchsanordnung wird auf den Homepages der beiden Web-Server jeweils eine Fliege eingebettet, und die Zugriffe auf die Homepage und die Fliege werden über einen Zeitraum von 24 Tagen gemessen. Die Zahlen werden tageweise anhand der Zugriffs-Protokolle bestimmt. Die Abbildung 4.4 zeigt die Zahl der täglichen Zugriffe auf die Homepage und auf die Fliege des DBIS-Servers. Abbildung 4.5 zeigt dieselben Daten für den Zeitungs-Server.

Bei beiden Web-Servern sind deutlich Zugriffseinbrüche an den Wochenenden zu verzeichnen. Beim DBIS-Server liegt der Grund dafür mit Sicherheit darin, daß die meisten Zugriffe von der Uni aus geschehen, welche am Wochenende geschlossen ist. Die Zeitung erscheint nur von Montag bis Samstag, weswegen sich zumindest die geringen Zugriffe am Sonntag erklären lassen. Warum ebenfalls der Samstag betroffen ist, muß andere Gründe haben und bleibt hier offen. Es legt jedoch die Vermutung nahe, daß häufig die am Arbeitsplatz vorhandenen Internet-Zugänge genutzt werden.

Die Zugriffe auf den Zeitungs-Server folgen einem periodischen Verlauf, während die Zugriffe auf den DBIS-Server stärkeren Schwankungen unterworfen sind.

Vergleicht man nun die Zugriffe auf die Fliegen auf den beiden Server mit denen auf die Homepage, so fällt der synchrone Verlauf der Fliege auf dem Zeitungs-Server sofort auf. Es deutet auf eine relativ konstante Auswirkung von Caching auf die Zugriffe hin. Die Zugriffe auf die Fliege auf dem DBIS-Server hingegen, weisen ein teilweise sogar asynchrones Verhalten auf, bzw. das Verhältnis der Zugriffe auf die Homepage und auf die Fliege sind starken Schwankungen unterworfen. In Abbildung 4.6 werden diese Verhältnisse auf den beiden Servern gegenübergestellt, wobei die Unterschiede deutlich ins Auge fallen. Berechnet werden diese Verhältnisse durch Division der Gesamtzugriffe auf die Fliege eines Tages durch die Anzahl der Gesamtzugriffe auf die Homepage desselben Tages.

Auf dem Zeitungs-Server liegt das Verhältnis zwischen Fliege und Homepage bei etwa $1,3$ mit geringen Schwankungen, während dieselben Verhältnisse auf dem DBIS-Server keinerlei konstante Auswirkung von Caching andeuten, d.h. starken Schwankungen unterliegen.

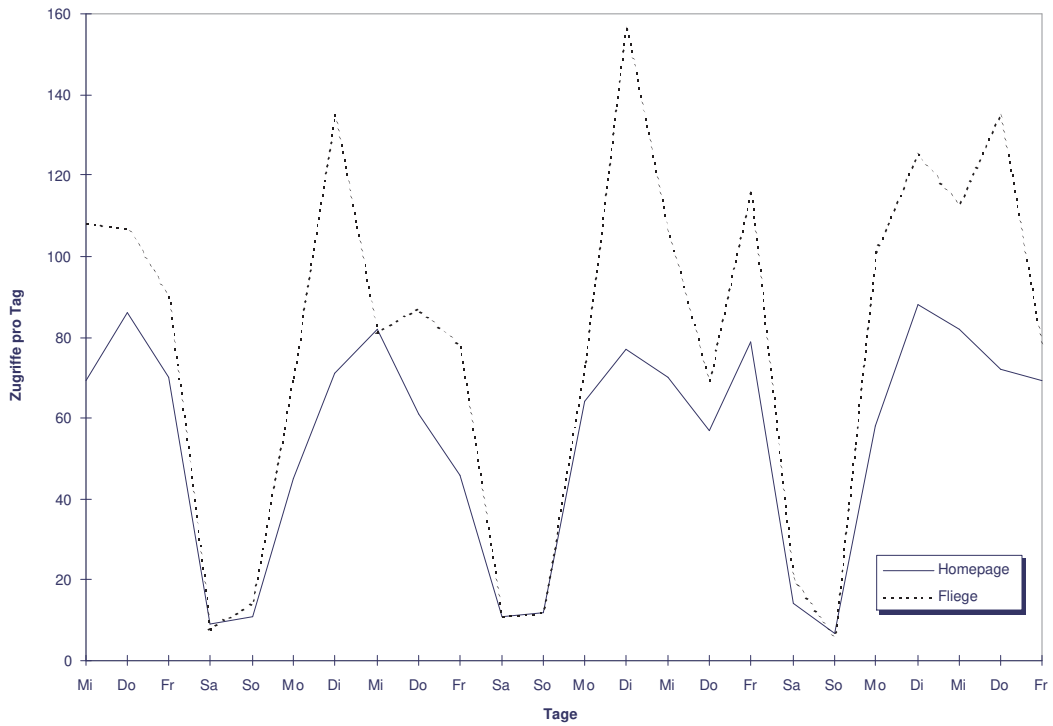

Abbildung 4.4: Zugriffe auf die Homepage und die Fliege des DBIS-Servers

4.4.3 Messungen der zweiten Versuchsanordnung

In der zweiten Versuchsanordnung werden zwei Web-Seiten auf dem Zeitungs-Server mit jeweils einer Fliege versehen, und die Zugriffe auf die Homepage und die Fliege werden über einen Zeitraum von 7 Tagen gemessen. Abbildung 4.7 zeigt die Zugriffe auf die Homepage und zugehörige Fliege (Fliege H), sowie die Zugriffe auf die zweite untersuchte Web-Seite und deren Fliege (Fliege S).

Der Verlauf der Zugriffe auf die Web-Seite ist, wie auch bei der Homepage, synchron. Wie in Abbildung 4.8 zu sehen ist, ist der Verlauf der Verhältnisse zwischen den Zugriffen auf die Web-Seite und auf die Fliege stärkeren Schwankungen unterworfen als bei der Homepage. Wichtig an dieser Versuchsanordnung war es, einmal zwei Web-Seiten derselben Umgebung zu untersuchen, da besonders die in Abschnitt 4.4.1 beschriebenen Eigenarten des DBIS-Servers einen direkten Vergleich der Ergebnisse erschweren. Außerdem sollten sich die untersuchten Web-Seiten in der Zahl der Gesamtzugriffe unterscheiden. Wie in Abbildung 4.7 ersichtlich, wird auf die Homepage wesentlich häufiger zugegriffen. Um den Verlauf der Zugriffe auf die Web-Seite besser deutlich zu machen, wird ein logarithmische Skala verwendet. Hier konnte also untersucht werden, ob sich das Verhalten des Verfahrens mit der Häufigkeit der Zugriffe auf eine Web-Seite ändert.

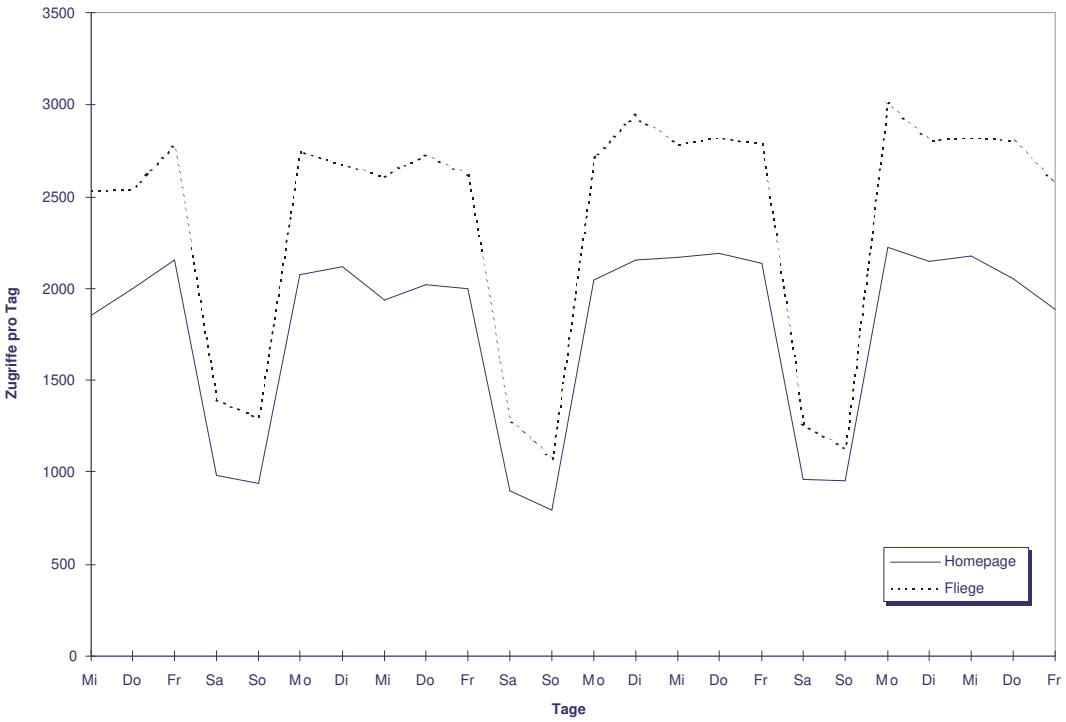

Abbildung 4.5: Zugriffe auf die Homepage und die Fliege des Zeitungs-Servers

4.4.4 Bewertung der Messungen

Theoretisch sollen die Zugriffe auf die Fliegen die tatsächlichen Zugriffe auf die Homepages widerspiegeln. Dadurch, daß bei den Messungen auf dem DBIS-Server aber teilweise weniger Zugriffe auf die Fliege als auf die Homepage gemessen wurden, muß dies zumindest auf diesem Server angezweifelt werden.
Die Schwankungen der Verhältnisse zwischen den Zugriffen auf die Fliege und die zugehörige Web-Seite sind bei allen drei untersuchten Seiten sehr unterschiedlich. Den geringsten Schwankungen unterliegen sie auf der Homepage des Zeitungs-Servers, danach kommt die zweite untersuchte Web-Seite des Zeitungs-Servers und zum Schluß die Homepage des DBIS-Servers mit den stärksten Schwankungen. Die Tatsache, daß diese Reihenfolge ebenfalls die Größenordnung der Zugriffe auf die entsprechende Web-Seite wiedergibt, ist ein Indiz dafür, daß die Stärke der Schwankungen mit der Anzahl der Anforderungen einer Web-Seite abnimmt. Dies müßte aber in einem umfangreicheren Versuch verifiziert werden, der mehrere Web-Seiten mit unterschiedlichen Zugriffscharakteristika über einen längeren Zeitraum untersucht.

4.5 Bewertung des Verfahrens

Die Verwendung nicht zwischenspeicherbarer Web-Dokumente kann, wie gezeigt wurde, die Auswertung der Protokolldateien (mit Rücksicht auf die genannten Schwierigkeiten)

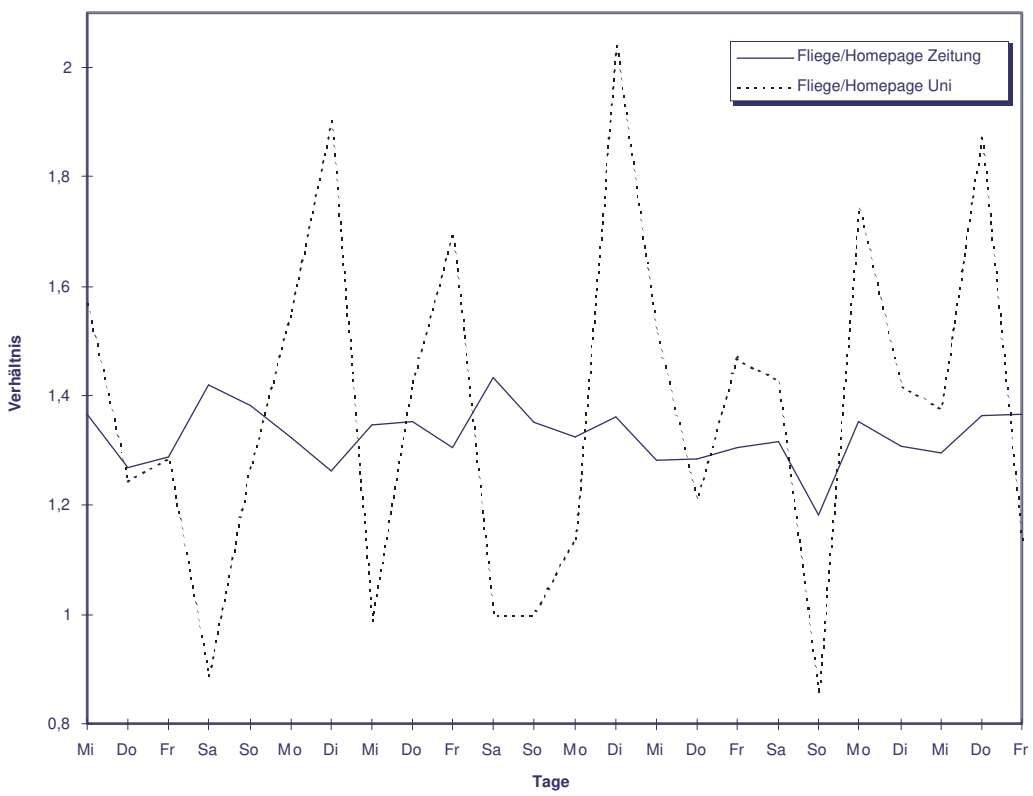

Abbildung 4.6: Verhältnisse der Zugriffe auf die jeweilige Fliege und Homepage auf den Servern

verbessern. Dabei darf man aber nicht außer acht lassen, daß die erzwungene Umgehung des Caching-Mechanismus der Idee des Caching entgegenwirkt, da eine unnötige Netzbelastung erzeugt wird.

Die Kosten für diese unnötige Übertragung trägt nicht derjenige (nämlich der Betreiber des Web-Servers) allein, der den meisten Nutzen von dieser überflüssigen Netzbelastung hat, sondern zum Teil auch der Benutzer, der die Web-Seite anfordert. Erfolgt die Netzanbindung zum Beispiel über eine Telefonleitung, so muß der Benutzer durch höhere Telefonkosten die Übertragung dieser Daten bezahlen.

Aus der Sicht des Betreibers des Web-Servers hat das Verfahren viele Vorteile. Zu den angesprochenen Korrekturmöglichkeiten, kann das CGI-Programm, das das Bild erzeugt, außerdem noch benutzt werden, um eine benutzerdefinierte Protokolldatei zu erstellen (s. Kapitel 2.3.3). Dabei entsteht zwar eine Mehrbelastung des Servers, diese geht allerdings nur zu Lasten des Betreibers und ist deswegen eigentlich nicht zu beanstanden. Die Laufzeit des CGI-Programms darf allerdings dadurch nicht so stark verlängert werden, daß der Benutzer wiederum durch verlängerte Wartezeiten benachteiligt wird. Dazu ist zu erwähnen, daß die Verbindung zwischen dem Browser und dem Server während der Laufzeit des CGI-Programms aufrecht erhalten wird. Dies kann durch geschickte Programmierung allerdings vermieden werden, indem die Verarbeitung der erhobenen Daten

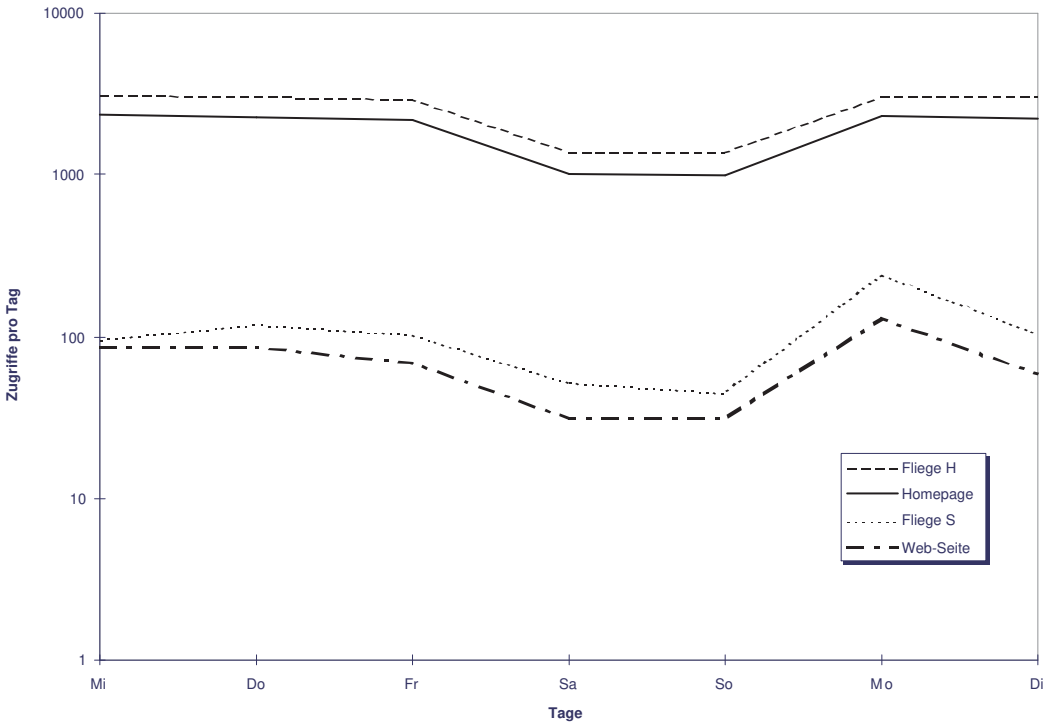

Abbildung 4.7: Zugriffe auf zwei Web-Seiten des Zeitungs-Servers und deren Fliegen

in einen externen Prozeß ausgelagert und die Verbindung zum Browser beendet wird.
Trotz aller Schwachpunkte stellt das Verfahren, zumindest von Seiten deutscher Anbieter, zur Zeit die praktikabelste Lösung dar. Die Gruppe Publikumszeitschriften des Verbandes Deutscher Zeitschriftenverleger (VDZ) (http://www.pz-online.de) hat sich auf ein Korrekturverfahren geeinigt, welches auf dem hier beschriebenen Prinzip beruht.

4.6 Vorschlag eines Alternativverfahrens

Das Verfahren kann sicherlich nützlich sein, um über einen gewissen Zeitraum festzustellen, in welcher Größenordnung der Einfluß von Caching auf dem untersuchten Server liegt. Es ist aber aus den genannten Gründen als permanentes Mittel zu Korrektur nur bedingt geeignet.
Eine Alternative für das Verfahren beruht auf der Erkenntnis aus den gemachten Messungen[1], daß man die Caching-Korrektur relativ gut durch einen konstanten Faktor annähern kann. Am Beispiel der Homepage des Zeitungs-Servers würde dieser bei etwa 1,3 liegen (siehe Abbildung 4.6 und 4.8).
Nachdem der Korrekturfaktor in einer Probemessung ermittelt wurde, kann auf die Fliege wieder verzichtet werden, bzw. muß diese nur noch in regelmäßigen Abständen zur Verifikation eingesetzt werden.

[1] Diese Annahme muß noch durch weitere Messungen verifiziert werden.

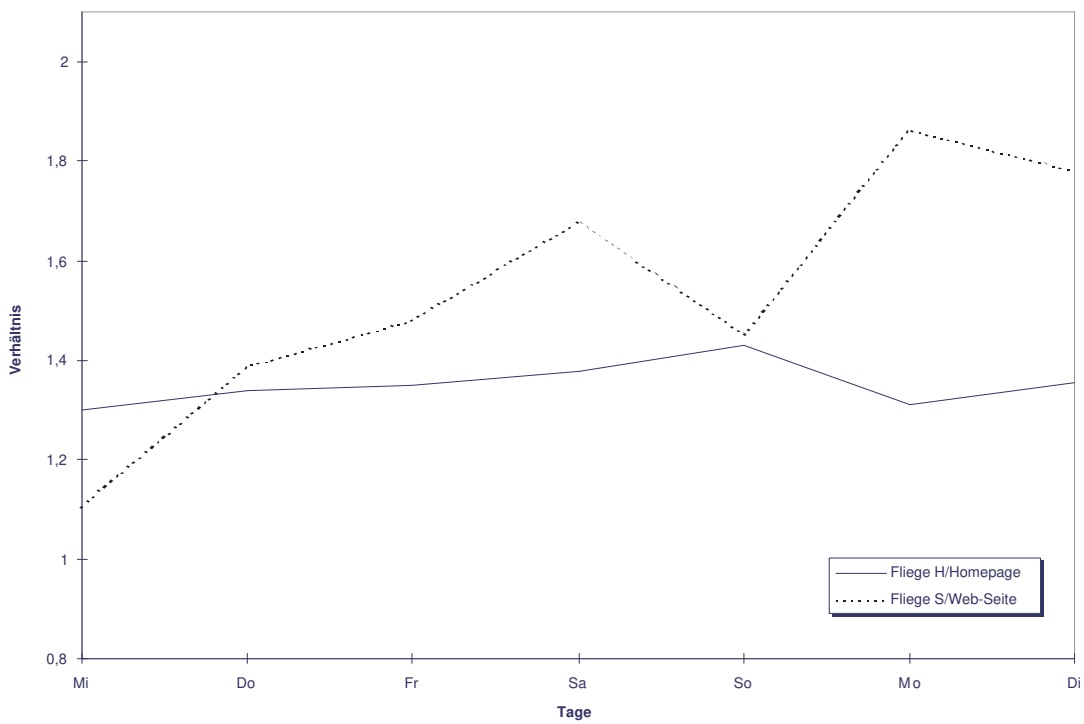

Abbildung 4.8: Verhältnisse der Zugriffe bei der zweiten Versuchsanordnung

Wie in Abschnitt 4.4.4 erläutert wird, ist ein relativ konstantes Verhältnis von Fliege und Homepage erst bei Web-Seiten mit hohen Zugriffszahlen zu erwarten. Daher wird ein brauchbarer Korrekturfaktor auch nur für diese Web-Seiten zu ermitteln sein. Bei Web-Seiten mit zu geringen Zugriffen sollte man daher prinzipiell auf die Verwendung der Fliegen verzichten. Dies ist aber ohne weiteres zu vertreten, da Web-Seiten mit derart geringen Zugriffszahlen auch nur von geringem wirtschaftlichen Interesse sind.

Kapitel 5

Bewertung und Ausblick

Durch die Kommerzialisierung des Internets werden immer neue Anforderungen an das Internet gestellt. Bei wachsendem Service wird gleichzeitig eine immer höhere Performanz gefordert. Um dem gerecht zu werden, wird die für die Vernetzung verwendete Hardware immer weiter ausgebaut und entwickelt und auch die möglichen Softwaretechniken, die die Leistung der Hardware unterstützen, werden stetig vorangetrieben. Eine zentrale Rolle nehmen hierbei die Caches ein, da sie die Netzbelastung unmittelbar senken und sich somit positiv auf die Netzleistung auswirken.

Die Caching-Technologie birgt aber nicht nur Vorteile. Die Geschwindigkeitssteigerung die erzielt wird, fordert ihre Tribute in Bereichen der Zugriffsprotokollierung und der Aktualität. Wie in dieser Arbeit gezeigt wird, resultieren die bekannten Lösungsmöglichkeiten für die genannte Problematik immer in einer Einschränkung der Geschwindigkeitssteigerung, welche durch Caching gewonnen wird.

Eine mögliche Lösung zur Korrektur von Zugriffszahlen, die Verwendung von nicht zwischenspeicherbaren Web-Dokumenten, ist nur durch den Mißbrauch von Techniken möglich ist, deren ursprünglicher Zweck nicht die Umgehung von Caching war. Dies weist auf Defizite hin, die das zugrundeliegende HTTP/1.0 Protokoll in bezug auf Caching hat. Der Entwurf von HTTP/1.1 macht deutlich welchen Stellenwert Caching heute hat. Eine direkte Unterstützung von Caching ist in den neuen Entwurf eingeflossen. Das bisherige Hilfsfeld (`pragma: no-cache`) wird durch eine eigene Cache-Kontrolle im HTTP-Header (siehe Abschnitt 2.1.4) ersetzt. Diese Neuerungen gehen weit über die Möglichkeiten hinaus, die man mit den Feldern im HTTP-Header von HTTP/1.0 in Bezug auf das Caching von Web-Dokumenten hat. Auffällig ist auch, daß der Empfehlungscharacter den die Behandlung von Caching in HTTP/1.0 hat, in HTTP/1.1 durchweg in Vorschriften geändert wird.

Für die kommerzielle Nutzung spielen die Protokolldateien eine zentrale Rolle, da mit ihnen Kundenverhalten und -akzeptanz bestimmt werden können. Für diesen Anwendungsbereich ist es somit wichtig, die Verfälschungen, wie sie in dieser Arbeit beschrieben worden sind, möglichst gut zu kompensieren. Die in diesem Bereich stark verbreiteten Web-Server von Netscape, Microsoft und O'Reilly machen durch die Einführung der erweiterten Protokolldateien deutlich, daß das Common-Logfile-Format den Bedürfnissen der Betreiber nicht mehr gerecht wird. Daß die erweiterten Protokolldateien keinem gemeinsamen Format folgen, ist zum einen wohl eine Folge des Konkurrenzkampfes. Es zeigt aber auch, daß hier noch Entwicklungsbedarf herrscht, um eine geeignete Repräsentation der erhobenen

Daten zu erzielen und zu standardisieren. Ein solches standardisiertes Protokolldateiformat ist außerdem für die Entwicklung Web-Server-unabhängiger Auswertungssoftware von Nutzen.
Das in Kapitel 4 vorgestellte Verfahren kann, hilfreich zur Korrektur von Zugriffszahlen sein. Erkauft wird dieser Informationsgewinn aber mit einer höheren Netzlast. Außerdem wird in dem Verfahren der Caching Mechanismus gezielt umgangen.
Als kritische Anmerkung sei hier auch gesagt, daß die Entscheidung ein sehr kleines, transparentes Bild in dem Verfahren zu verwenden, vermutlich nicht nur der Minimierung der Netzlast dient, sondern auch einen unsichtbaren Betrieb des Verfahrens ermöglichen soll. Der Benutzer wird also im unklaren gelassen, daß die Zugriffe auf die Web-Seite in jedem Falle registriert werden. Außerdem müssen die Fliegen nicht notwendigerweise auf den Web-Servern erzeugt werden, auf deren Web-Seiten sie eingebettet werden. Ein Verfolgen eines Benutzers über die Grenzen eines Web-Servers hinweg wird dadurch ebenfalls vereinfacht.
Der Umfang der in dieser Arbeit vorgenommenen Messungen ist zu gering, um die aufgestellten Behauptungen zu untermauern. Besonders die Vermutung, daß bei hinreichend großen Zugriffszahlen ein konstanter Zugriffsfaktor angenommen werden kann, muß durch größer angelegte Messungen untermauert werden. Wenn diese Annahme bestätigt wird, kann das vorgestellte Alternativverfahren verwendet werden. Dadurch können bei geringeren Netzbelastungen vergleichbare Korrekturen vorgenommen werden.
Weiterhin ist ein genaue Untersuchung des Image-Off Faktors nötig, da hierin das Hauptproblem bei der Korrektur mit nicht zwischenspeicherbaren Web-Dokumenten liegt.
Die Korrekturverfahren müssen weiter beobachtet werden, da sich mit der Einführung von HTTP 1.1 neue Möglichkeiten zur Caching-Kontrolle ergeben, die der Leistungsverbesserung der geschilderten Verfahren dienlich sein werden.

Literaturverzeichnis

[ASA+95] M. Abrams, C. R. Stanridge, G. Abdulla, S. Williams, and E. A. Fox, editors. *Caching Proxies: Limitations and Potentials*, Proceedings of the Fourth International World Wide Web Conference, 1995.

[BLFF95] T. Berners-Lee, R. Fielding, and H. Frystyk. Hypertext Transfer Protocol - HTTP/1.0. http://info.cern.ch/hypertext/WWW/Protocols/HTTP/HTTP.html, October 1995.

[BMV96] H. Bekker, I. Melve, and T. Verschuren. Analysis of the functionality and effectiveness of Web caching servers. http://sunsite.icm.edu.pl/~wojsyl/desire/del_m_41.html, 1996.

[Bor95] G. Born. *Noch mehr Dateiformate*. Addison-Wesley, 1995.

[CDN+96] A. Chankhunthod, P. B. Danzig, C. Neerdaels, M. F. Schwartz, and K. J. Worrell. A Hierarchical Internet Object Cache. http://catarina.usc.edu/danzig/cache.ps, 1996.

[Con95a] WWW Consortium. HTML 2 Specification. http://www.w3.org/pub/WWW/MarkUp/html-spec_toc.html, 1995.

[Con95b] WWW Consortium. HTTP Status codes. http://www.w3.org/pub/WWW/Protocols/HTTP/HTRESP.html, 1995.

[Con95c] WWW Consortium. Logging Control in W3C httpd. http://www.w3.org/pub/WWW/Daemon/User/Config/Logging.html, 1995.

[Con96a] WWW Consortium. HTML 3 Specification. http://www.w3.org/pub/WWW/html3/Contents.html, 1996.

[Con96b] WWW Consortium. HTML 3.2 Specification. http://www.w3.org/pub/WWW/TR/WD-html32, 1996.

[Cor95a] Netscape Communications Corporation. Documentation for Netscape Enterprise Server 2.0. http://home.netscape.com, 1995.

[Cor95b] Netscape Communications Corporation. Persistent Client State - HTTP-Cookies. http://home.netscape.com/newsref/std/cookie_spec.html, 1995.

[Cor96] A. Cormack. Web Caching. http://www.niss.ac.uk/education/jisc/acn/caching.html, 1996.

[CZ95] D. B. Chapman and E. D. Zwicky. *Building Internet Firewalls.* O'Reilly & Associates, Inc., 2nd. edition, 1995.

[ea95] B. Barron et al., editor. *The Internet 1996*, chapter 2. Sams.net Publishing, 1995.

[FGM+95] R. Fielding, J. Gettys, J. C. Mogul, H. Frystyk, and T. Berners-Lee. Hypertext Transfer Protocol - HTTP/1.1. draft-ietf-http-v11-spec-06, August 1995.

[Gol95] J. Goldberg. Why web usage statistics are (worse than) meaningless. http://www.cranfield.ac.uk/stats/, 1995.

[HSW96] D. R. Hardy, M. F. Schwartz, and D. Wessels. *Harvest User's Manual.* University of Colorado at Boulder, Department of Computer Science, University of Colorado, Boulder, Colorado 80309-0430, January 1996.

[KL86] B. Kantor and P. Lapsley. Network News Transfer Protocol (RFC977), February 1986.

[LRV96] P. Lorenzetti, L. Rizzo, and L. Vicisano. Replacement policies for a proxy cache. http://www.iet.unipi.it/uigi/caching.ps.gz, 1996.

[LS87] P.C. Lockemann and J.W. Schmidt, editors. *Datenbank-Handbuch.* Springer Verlag, 1987.

[MLB95] R. Malpani, J. Lorch, and D. Berger, editors. *Making World Wide Web Caching Servers Cooperate*, Proceedings of the Fourth International World Wide Web Conference, 1995.

[Mor95] C. Morrow, editor. *HTML & CGI unleashed.* Sams.net Publishing, 1995.

[Pos85a] J. B. Postel. File Transfer Protocol (RFC959), October 1985.

[Pos85b] J. B. Postel. Simple Mail Transfer Protocol (RFC821), 1985.

[Sch95] O. Schade. Hierarchischer Cache-Verbund mit Harvest. *iX*, 8:124–129, August 1995.

[Ste94] W. R. Stevens. *TCP/IP Illustrated Volume 1.* Addison Wesley, 1994.