Web-Caching

Seminararbeit zur Veranstaltung
Verteilte und parallele Systeme II
bei Prof. Dr. Rudolf Berrendorf
im Wintersemester 04/05

Von Mark Dittmann, Markus Lepper, Frank Waibel

Fachbereich Informatik
Fachhochschule Bonn-Rhein-Sieg, St. Augustin

Inhaltsverzeichnis

1 Einleitung und Motivation 3
2 Das Konzept des Web-Caching 3
2.1 Aufgabe eines Web-Caches 4
2.2 Arten von Web-Caches 5
2.3 Bewertung von Web-Caches 6
2.4 Web-Pre-Caching 8
3 Verfahren zur Garantie der Datenaktualität 10
3.1 Unterstützung durch HTTP/1.0 11
3.2 Unterstützung durch HTTP/1.1 13
3.3 Steuerung des Web-Cache-Verhaltens 15
4 Web-Caching in lokalen Netzwerken 16
4.1 Proxy-Caching-Lösung 16
4.2 Transparente Caching-Lösung 18
5 Verbundsysteme von Web-Caches 18
5.1 Funktionsweise eines Web-Cache-Verbundsystems 19
5.2 Kooperatives Web-Caching 20
6 Das ICP-Protokoll 20
6.1 Konfiguration von ICP 21
6.1.1 ICP nur zwischen Siblings 21
6.1.2 ICP zwischen Siblings und Parents 22
6.2 Funktion von ICP 23
7 Zusammenfassung und Ausblick 24
8 Quellenverzeichnis 25

1 Einleitung und Motivation

Laut einer in der Informatik weit verbreiteten Regel neigen Internet-Benutzer – insbesondere innerhalb einer Organisation (z.B. Universität, Unternehmen) – dazu, auf Ressourcen, die sie bereits verwendet haben, innerhalb kurzer Zeit erneut zuzugreifen.
Aus diesem Grunde ist es sinnvoll, ein aus der Informatik bekanntes und bewährtes Konzept, das Caching, auch in diesem Bereich einzusetzen.
In der lokalen Umsetzung dieses Prinzips werden die von einem Rechner bereits abgerufenen Ressourcen in dem Browser-Cache über eine gewisse Zeit für eventuelle erneute Abrufe gespeichert, so dass die Ressourcen nicht mehrfach über das Netz transferiert werden müssen.

Dieses Caching-Prinzip wurde auch für Web-Dienste übernommen. Die Intention war, die Übertragungsnetze zu entlasten, Übertragungskosten einzusparen und gewünschte Web-Inhalte schneller liefern zu können. Diese Form des Caching bezeichnet man als Web-Caching.

Für die Realisierung eines Web-Caches wird in der Regel ein spezieller Rechner, ein so genannter Web-Cache-Server, eingesetzt. Inzwischen sind diese Web-Cache-Server sowohl für viele Unternehmen als auch für die Internet Service Provider (ISP) unverzichtbar geworden. Die großen ISPs setzen nicht nur einen Web-Cache-Server, sondern ein ganzes Netzwerk von Servern ein. Dieses vernetzte System von Web-Cache-Servern, in dem die einzelnen Server miteinander kooperieren, wird als Web-Caching-System bezeichnet.

Hier ein kurzer Überblick darüber, welche Fragen in dieser Seminararbeit geklärt werden:

- Wie funktioniert Web-Caching und welche Arten von Web-Caches gibt es?
- Welche Vorteile ergeben sich durch das Web-Caching?
- Wie wird die Aktualität des Content in Web-Caches überprüft?
- Wie kann Web-Caching in lokalen Netzwerken eingesetzt werden?
- Wie kann ein vernetztes Web-Caching-System aufgebaut werden?

2 Das Konzept des Web-Caching

Grobe Schätzungen besagen, dass ca. 10% aller Web-Seiten weltweit derart populär sind, dass sie ca. 90% aller Anfragen erhalten.
Aus diesem Grunde sind gerade diese sehr beliebten Web-Server stark überlastet. Web-Caches können diese Überlastung mindern, da nicht mehr so viel Bandbreite dafür verschwendet wird, denselben Web-Content an verschiedene Benutzer zu übertragen. Das Web-Caching sorgt nicht nur für

eine Entlastung der stark frequentierten Web-Server, sondern auch für eine Performance-Verbesserung, da die Übertragung vom schnellen Web-Cache weniger Zeit in Anspruch nimmt als die Übertragung vom überlasteten Web-Server.

2.1 Aufgabe eines Web-Caches

Die Aufgabe eines Web-Caches besteht im Wesentlichen darin, einmal abgerufene Web-Objekte (Texte, Grafiken, Videos und andere Daten) an einer Lokation in der Nähe des Benutzers abzuspeichern. Wenn der Benutzer nun erneut oder ein anderer Benutzer zum ersten Mal auf eine bereits abgerufene Web-Seite zugreift, werden die angeforderten Objekte vom Web-Cache geliefert und nicht mehr vom eigentlichen Web-Server.

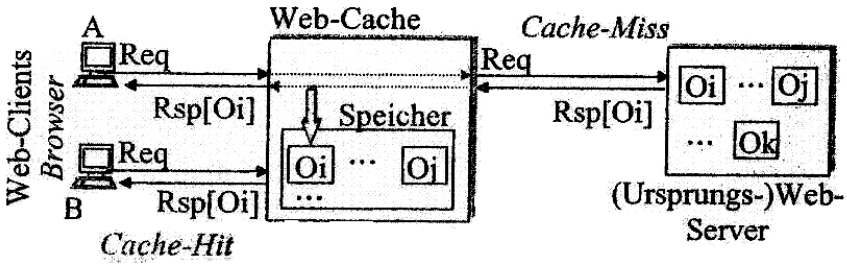

Abbildung 1: Aufgabe eines Web-Caches

Wie man in Abbildung 1 erkennen kann, wird der Web-Cache irgendwo zwischen Web-Client und Web-Server installiert, aber möglichst nahe am Web-Client. Der Web-Cache-Server muss ein Rechner mit viel Speicherkapazität, einer schnellen Netzanbindung und einem besonders I/O-Verhalten sein.
Um einen lokalen Netzwerk-Web-Cache (z.B. in einem Unternehmen oder einer Hochschule) nutzen zu können, muss der Browser so konfiguriert werden, dass alle HTTP-Requests zuerst an den Cache-Server gerichtet werden. Der Cache-Server prüft dann, ob sich das angeforderte Web-Objekt in seinem Speicher befindet. Wenn dies nicht der Fall ist, bezeichnet man das als **Cache-Miss**. In diesem Fall leitet der Web-Cache den betreffenden HTTP-Request an den Web-Server weiter, der dann das gewünschte Objekt an den Web-Cache liefert, welcher es dann an den Browser weiterleitet und gleichzeitig eine Kopie des Objekts in seinem Speicher anlegt.
Im Falle, dass ein angefordertes Objekt bereits im Speicher des Cache-Servers vorhanden ist, spricht man von einem **Cache-Hit**. Der Cache-Server liefert dann direkt das Objekt an den Browser, ohne den Request an den Web-Server weiterzuleiten. Dies spart Traffic (und damit Kosten) und entlastet den Web-Server. Ein Web-Cache-Server verhält sich gegenüber Web-Servern genau wie ein Browser, so dass der Web-Server keinen Unterschied erkennen kann.
Allerdings ist die Aufgabe eines Web-Caches doch etwas komplizierter als das einfache Speichern von HTTP-Responses und deren Weitergabe bei Anforderung. Zusätzlich muss ein Mechanismus implementiert werden, der

überprüfen kann, ob der gewünschte Web-Content noch aktuell ist. Auf diese Thematik wird näher in Abschnitt 3 eingegangen.
Ein Web-Cache ist immer durch seinen Speicher begrenzt. Wenn dieser voll ist, muss der Web-Cache mittels einer Verdrängungsstrategie (Cache Replacement Policy) dafür sorgen, dass neuer Content gespeichert werden kann.
Mögliche Verdrängungsstrategien:

- **LRU** (Least Recently Used): Content, der am längsten nicht mehr benötigt wurde, wird gelöscht.
- **LFU** (Least Frequentlich Used): Content, auf den am seltensten zugegriffen wurde, wird gelöscht
- Andere: **SLRU** (Segmented LRU) und **GDS** (Greedy Dual Size)

2.2 Arten von Web-Caches

Web-Caches unterscheiden sich je nach ihren unterschiedlichen Ausprägungen und Einsatzorten:

- Browser-Caches,
- Netzwerk-Caches in Unternehmen (Universitäten, etc.),
- Web-Caches bei Internet Service Providern (ISPs) und
- WebCaches bei Content-Anbietern (sog. Reverse-Caches).

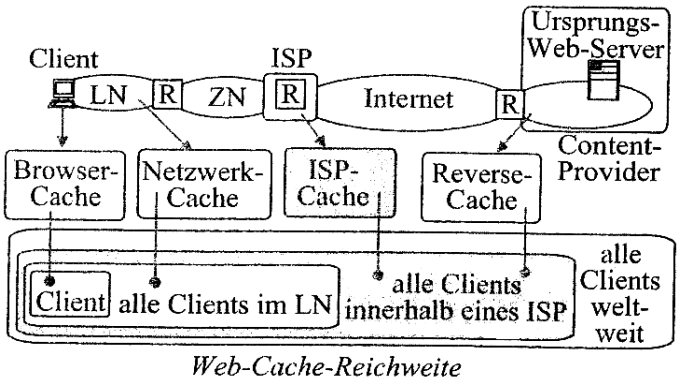

Abb. 2: Arten von Web-Caches [LN=Lokales Netz, ZN=Zugangsnetz, R=Routerf.]

Browser-Caches:
Heutzutage hat jeder Web-Browser einen einfachen integrierten Cache, der bereits abgerufenen Web-Content lokal im Rechner für einen eventuellen erneuten Zugriff abspeichert.

Netzwerk-Caches:
Ein Netzwerk-Cache ist ein Cache-Server, der abgerufenen Web-Content eines gesamten Netzwerks zwischenspeichert und bei erneuter Anforderung durch denselben oder einen anderen Nutzer des Netzwerks liefert. Ein Netzwerk-Cache-Server wird meist vor der Internet-Anbindung des Netzwerks installiert.

Web-Caches bei Internet Service Providern:

Die meisten ISPs haben spezielle Web-Cache-Systeme installiert, um ihre Kunden schneller mit Web-Inhalten versorgen zu können und gleichzeitig Traffic und damit Kosten einzusparen. Idealerweise liegt ein solches Web-Cache-System am so genannten Point of Presence (PoP).

Reverse-Caches (Surrogates):
Cache-Server, die direkt bei großen Anbietern von Web-Content eingesetzt werden, haben die Aufgabe, die Belastung von Web-Servern zu verringern, indem sie statischen Web-Content vorhalten, und die Reaktionsgeschwindigkeit der Web-Server zu erhöhen

Wie in Abbildung 2 dargestellt wird, können sich auf der Strecke zwischen Client und Web-Server eine Reihe von Web-Caches befinden. Je näher ein Web-Server an dem Ursprungs-Web-Server ist, desto größer ist die Anzahl der Clients, die er bedienen kann. Dies wird mit dem Begriff **Web-Cache-Reichweite** bezeichnet.

2.3 Bewertung von Web-Caches

Wenn man eine Bewertung von Web-Caches vornehmen will, muss man sich an den bereits erwähnten Cache-Hits und Cache-Misses orientieren.
Abbildung 3 stellt diese beiden Fälle gegenüber.

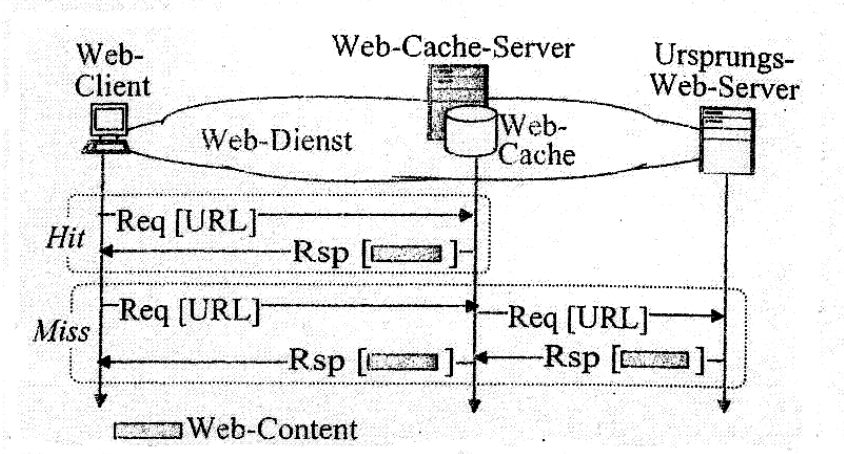

Abb. 3: Gegenüberstellung Cache-Hit und Cache-Miss

Je öfter Cache-Hits vorkommen, desto effektiver arbeitet ein Web-Cache und deshalb kann anhand der Cache-Hit-Häufigkeit eine Aussage über die Cache-Effizienz gemacht werden. Den Anteil der Hits an den eingehenden Requests bezeichnet man als **Cache-Hit-Rate**.
Im Gegenzug ist es logisch: je häufiger Cache-Misses vorkommen, desto uneffektiver arbeitet der Web-Cache. Das Verhältnis von Cache-Misses zu allen Requests bezeichnet man analog als Cache-Miss-Rate.

Zur Ermittlung der Effizienz des Web-Caches, wird er als Vermittler zwischen Web-Clients und Web-Servern wie in Abbildung 4 dargestellt.

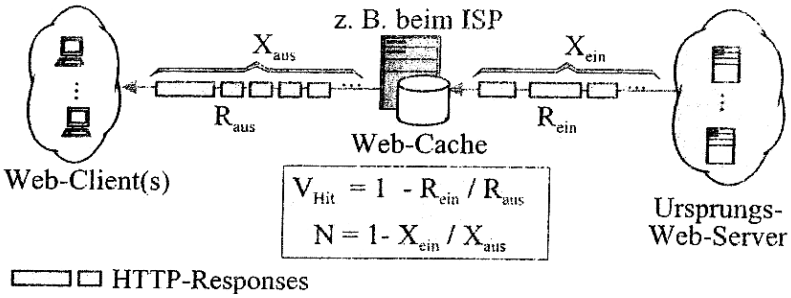

Abb. 4: Hit-Rate und Nutzbarkeit eines Web-Caches

- R_{ein} die Anzahl der eingehenden HTTP-Responses von Web-Servern (damit = Anzahl der Misses),
- R_{aus} die Anzahl der ausgehenden HTTP-Responses zu Web-Clients,
- Anzahl von Hits gleich $R_{aus} - R_{ein}$.
- Hit-Rate: V_{Hit} [%] = ($R_{aus} - R_{ein}$) / R_{aus} = 1 – R_{ein} / R_{aus} [100%]
- Miss-Rate: V_{Miss} [%] = $R_{ein} - R_{aus}$ [100%]

Die Hit-Rate V_{Hit} kann sowohl Seiten-Hit-Rate als auch als Objekt-Hit-Rate interpretiert werden und liegt typischerweise im Bereich zwischen 35% und 60%. Je größer die Datenmenge ist, die ein Web-Cache aus seinem Speicher unmittelbar an die Web-Clients liefert, desto größer ist seine Nutzbarkeit. Die Nutzbarkeit lässt sich mit folgenden Werten Berechnen:

- X_{ein} die Anzahl von Bytes aus allen eingegangen HTTP-Responses von Web-Servern und
- X_{aus} die Anzahl von Bytes aller ausgegangenen HTTP-Responses zu Web-Clients,

$$N\ [\%] = (X_{aus} - X_{ein}) / X_{aus} = 1 - X_{ein} / X_{aus}\ [100\%]$$

Die **Cache-Nutzbarkeit** (Cache Utilization) liegt oft im Bereich zwischen 20% und 40% und kann zur Berechnung der Einsparungen an Übertragungskosten zwischen Web-Cache und Web-Servern genutzt werden.
Wenn die Nutzbarkeit eines ISP-Caches bekannt ist, so kann die Ersparnis bzw. der Gewinn direkt berechnet werden.

Prinzipiell kann man feststellen, dass je größer der Speicher eines Web-Caches ist, desto höher ist die Hit-Rate und damit die Nutzbarkeit. Aus diesem Grunde verwenden ISPs große Web-Cache-Systeme, die eine enorme Speicherkapazität haben.

Experimentelle Untersuchungen beweisen zusätzlich Folgendes:

- Die Hit-Rate eines Web-Caches wächst ungefähr logarithmisch mit der Zunahme der Anzahl von Web-Clients, die er mit Content versorgen kann. Dies bedeutet, dass die Hit-Rate eines Web-Caches umso größer wird, je größer seine Reichweite ist.

- Je größer die Reichweite eines Web-Caches ist, desto größer ist auch die gesamte Anzahl von Anfragen. Daraus folgt, dass die Hit-Rate des Web-Caches auch ungefähr logarithmisch mit der Zunahme von Anfragen wächst.

Zu bemerken ist allerdings, dass mit Vergrößerung der Reichweite ein Verlust an Geschwindigkeits-Performanz verbunden ist. Eine Aussage über die Performanz eines Web-Caches lässt sich über die mittlere Reaktionszeit und über den Cache-Durchsatz, der durch die maximale Anzahl der verarbeiteten http-Responses pro Sekunde bestimmt wird, machen.

Der Durchsatz und die Reaktionszeit eines Web-Cache-Servers werden größtenteils durch dessen I/O-Leistung geprägt. Insbesondere die verwendeten Speichermedien, aber auch das Betriebssystem, haben daran maßgeblichen Anteil. Auch die Größe und Geschwindigkeit des Arbeitsspeichers, in dem besonders häufig frequentierte Web-Objekte abgelegt werden, nimmt einen entscheidenden Einfluss.

2.4 Web-Pre-Caching

Durch die immer häufiger vorkommende Verteilung von Inhalten einer einzigen Web-Seite auf mehrere Web-Server und dort abgelegte Web-Ressourcen (z.B. durch Links oder eingefügte Bilder) ist das traditionelle Caching einer einzelnen Web-Ressource immer weniger effizient. In neuen Ansätzen wird daher versucht, Web-Inhalte, auf die innerhalb der aktuell angezeigten Web-Seite verwiesen wird, schon im Voraus abzurufen und im Cache zu speichern, bevor der Nutzer diese anwählt. Dieses wird als **Pre-Caching** bezeichnet.

Mit **Web-Pre-Caching** oder auch **Content-Pre-Fetching** bezeichnet man jegliche Art von Web-Caching, bei der selbstständig Ressourcen aus dem Web abgerufen werden, um diese Nutzern bei eventuellem Request sofort „liefern“ zu können.
Abbildung 5 stellt das Verhalten eines Web-Caches beim Web-Pre-Caching dar:

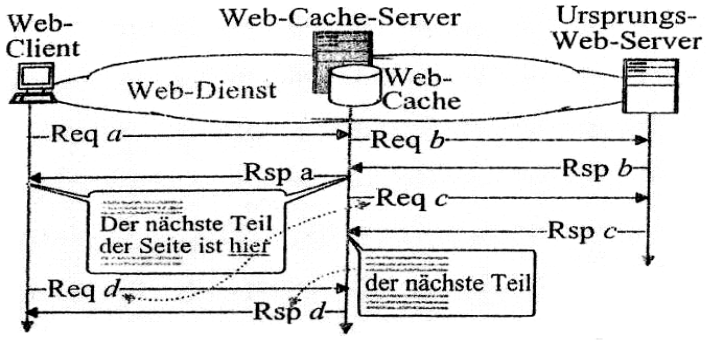

Abb. 5: Web-Pre-Caching in einem Web-Cache-Server

1. Web-Client sendet HTTP-Request Req a.
2. Gewünschter Web-Content ist nicht im Cache verfügbar.
3. Web-Cache-Server ruft den Content vom Web-Server mit Req b.
4. Web-Server sendet den Content im HTTP-Response Rsp b.
5. Rsp b wird im Cache abgelegt und an den Client weitergegeben.
6. Web-Cache erkennt Link in Rsp b auf weiteres Web-Objekt.
7. Web-Cache-Server ruft den Content vom Web-Server mit Req c.
8. Web-Server sendet den Content mit Rsp c.
9. Wenn der Client mit Req d den Content aus dem Link in Rsp a anfordert, kann der Cache diesen sofort liefern. Dadurch verringert sich die Wartezeit für den Client.

Web-Pre-Caching kann, ähnlich wie Web-Caching im Allgemeinen, an verschiedenen Orten stattfinden:

- beim Web-Client,
- beim Web-Proxy bzw. dessen Web-Cache oder
- beim Edge-Server eines Content Delivery Networks.

Web-Client:
Ein Web-Client kann Web-Pre-Caching realisieren, indem z.B. – ohne dass der Nutzer es merkt – die Inhalte von Links auf der aktuell angezeigten Web-Seite im Hintergrund abgerufen und im lokalen Web-Cache abgelegt werden. Das Problem bei dieser Form des Pre-Caching besteht darin, dass im Falle, dass der im Voraus gecachte Content nicht benötigt wird, unnötig viele Übertragungskapazitäten genutzt wurden.

Web-Proxy:
Das Pre-Caching im Proxy funktioniert auf die gleiche Weise wie beim Web-Client. Der einzige Unterschied besteht darin, dass der Nutzer nicht das Risiko unnötigen Traffics eingeht und dass nicht nur ein Nutzer von dem Pre-Caching profitieren kann, sondern auch alle anderen Benutzer, die über den gleichen Web-Proxy diese Seite abrufen.

Edge-Server / Content Delivery Network:
Das Prinzip des Pre-Caching kann auch in Content Delivery Networks zum Einsatz kommen. Dort kann dann ein so genannter **Edge-Server**, der als Reverse Cache fungiert, Web-Seiten oder anderen Web-Content im Voraus im Cache ablegen, die erfahrungsgemäß von den Nutzern häufig angefordert werden. Dies kann auch in einem festen Zyklus passieren, so dass der Content immer auf einem aktuellen Status ist.

Bei der Realisierung des Web-Pre-Caching wird häufig die Pipelining Funktion des HTTP/1.1 verwendet, um parallel Inhalte von mehreren Links in einer Web-Seite im Voraus abzurufen.

Es gibt eine Vielfalt von anderen Möglichkeiten des Pre-Caching, aber es lässt sich generell festhalten, dass ein intelligentes Verfahren im Voraus

Daten abruft und in den Cache speichert, die mit großer Wahrscheinlichkeit von einem, mehreren oder einer großen Menge von Nutzern abgerufen werden. Dies können entweder häufig frequentierte oder auch vermutlich in naher Zukunft angeforderte Daten sein.

3 Verfahren zur Garantie der Datenaktualität

Wie bei jeder Art von Caching-Systemen ist selbstverständlich auch bei Web-Caches die Aktualität des Cacheinhaltes fundamental wichtig für den effizienten und sinnvollen Einsatz. Es müssen also Verfahren gefunden und eingesetzt werden, die in der Lage sind die Aktualität des zwischengespeicherten Web-Contents zu überprüfen und im Bedarfsfall die Daten zu verwerfen und neu von Ihrer Quelle zu laden.

Aufgrund der Tatsache, dass die **Content-Validierung** bei jedem Request durchgeführt werden muss, unterscheidet man zwischen unbestätigten und bestätigten Cache-Hits.
Ebenso unterscheidet man zwischen unbestätigten und bestätigten Cache-Misses.

Unbestätigter Cache-Hit:
Der gewünschte Web-Content befindet sich bereits im Speicher des Web-Caches. Der Web-Cache kann die Content-Validierung selber vornehmen und muss die Aktualität des Content nicht vom Ursprungs-Web-Server bestätigen lassen. In diesem Fall erhält der Client den gewünschten Content am schnellsten.

Bestätigter Cache-Hit:
Der gewünschte Web-Content befindet sich bereits im Speicher des Web-Caches. Allerdings kann der Cache die Content-Validierung nicht selbst durchführen und muss die Content-Aktualität durch den Ursprungs-Web-Server bestätigen lassen. Der Ursprungs-Web-Server bestätigt die Aktualität des Content mit dem HTTP-Response `304 not modified`. Dann kann der Web-Cache-Server den Content aus seinem Speicher an den Web-Client ausliefern.

Bestätigter Cache-Miss:
Der gewünschte Web-Content befindet sich bereits im Speicher des Web-Caches. Bei der Content-Validierung durch den Ursprungs-Web-Server meldet dieser, dass der betreffende Content nicht aktuell ist und sendet die aktuelle Fassung des Contents an den Web-Cache-Server. Aus Client-Sicht beansprucht dieser Fall die höchste Wartezeit.

Unbestätigter Cache-Miss:
Der gewünschte Web-Content befindet sich nicht im Speicher des Web-Cache-Servers. Es folgt das Standard-Vorgehen eines Web-Caches.

Diese 4 verschiedenen Fälle machen deutlich, dass ein Web-Cache nur dann einen wesentlichen Vorteil erbringt, wenn die Anzahl der Cache-Hits überwiegt, da er im Falle von Cache-Misses die Antwortzeiten sogar entscheidend verlängert.

Zur Validierung des Web-Content gibt es mehrere Möglichkeiten, die jeweils auf dem HTTP-Protokoll aufsetzen. Diese sollen im Folgenden erläutert werden.

3.1 Unterstützung durch HTTP/1.0

In der Spezifikation des Hypertext Transfer Protocol (**RFC 1945**) sind explizite Mechanismen definiert, die Caching unterstützen und eine möglichst hohe Effizienz anstreben. So sollen in erster Linie Anfragen an Ursprungswebserver vermindert und unnötige Datenübertragungen verhindert werden.

Bei HTTP/1.0 werden dazu bestimmte **Header-Zeilen** verwendet:

- `Last-Modified` (im Entity-Header)
- `Expires` (im Entity-Header)
- `If-Modified-Since` (im Request-Header)

Das Feld `Last-Modified` beinhaltet den Zeitpunkt der letzten Änderung des Webinhaltes aus Sicht des Senders.

Zum Beispiel:

```
Last-Modified: Sat, 28 Dec 2002 16:11:34 GMT
```

Anhand dieser Zeile kann ein Web-Cache feststellen, ob er den angeforderten Inhalt direkt seinem Cache entnehmen kann oder ob er ihn erneut von dem Ursprungs-Server abrufen muss. Hierbei muss jedoch beachtet werden, dass diese Angabe unter Umständen nicht der letzten Veränderung entsprechen muss, da diese Funktion je nach Webserver unterschiedlich implementiert werden kann. So kann dieses Feld bei dynamisch erzeugten Webseiten den Zeitpunkt der Generierung enthalten oder bei Datenbankanfragen den Zeitstempel der Abfrage.

Die Zeile `Expires` gibt den Zeitpunkt an, an dem der Inhalt des zugehörigen Web-Contents nicht mehr gültig ist. Bis zu diesem Moment kann der Content gefahrlos gespeichert werden.

Beispiel: `Expires: Sat, 28 Dec 2002 16:11:34 GMT`

Nach diesem Zeitpunkt muss ein Web-Cache diesen Inhalt erneut vom Ursprungs-Webserver abrufen. Es gibt auch Inhalte die generell nicht von Web-Caches zwischengespeichert werden dürfen. Zu dieser Kategorie sind beispielsweise dynamisch erzeugte Inhalte zu zählen. Sie enthalten in der `Expires`-Zeile die gleiche Angabe wie in der Zeile `Date`.

Auch über die Zeile `If-Modified-Since` kann Web-Content von einem Server angefragt werden. Nur wenn sich der Inhalt der Ressource seit dem mitgegebenen Datum geändert hat, werden die Daten erneut übertragen.

Beispielsweise:

```
If-Modified-Since: Sat, 28 Dec 2002 16:11:34 GMT
```

Dieser Mechanismus kann sinnvoll eingesetzt werden, wenn ein Webcache eine lokale Kopie einer Ressource zwischenspeichert, die angefragt wurde, deren Gültigkeit jedoch über `Expires` abgelaufen ist. In diesem Fall kann der Web-Cache über die Zeile `If-Modified-Since` verifizieren, ob der Inhalt auch weiterhin aktuell ist und gegebenenfalls eine Neuübertragung veranlassen. Dies wird in der folgenden Grafik noch einmal verdeutlicht:

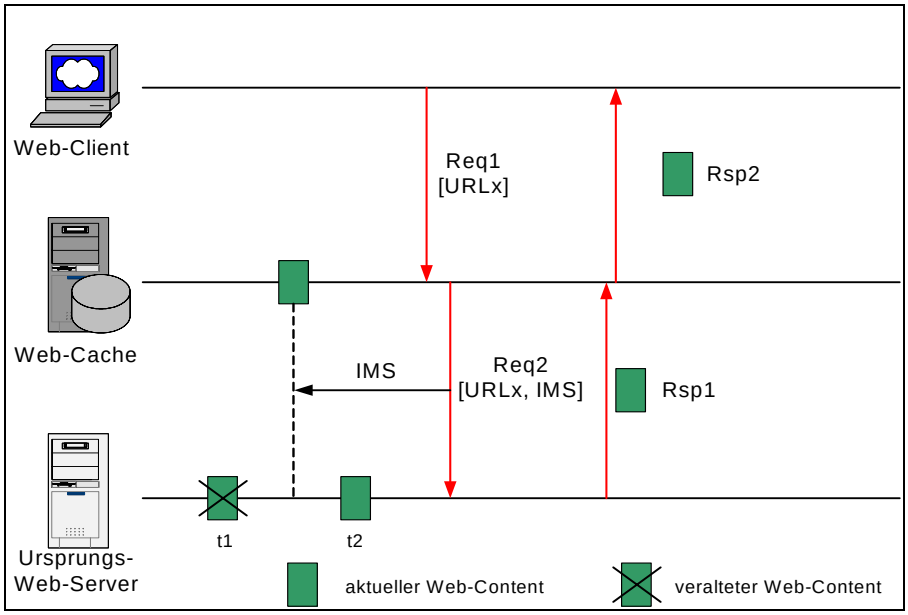

Abb.6: Anwendung von If-Modified-Since

Hier hat ein Web-Client die Ressource `URLx` angefragt (`Req1`). Der Web-Cache hält eine Kopie des angeforderten Content von dem Zeitpunkt `t1` in seinem Speicher. Nun verifiziert er die Aktualität des Contents mit einer `If-Modified-Since` Anfrage (`Req2`) bei dem Ursprungs-Web-Server und erhält von dem diesem die aktuellste Version der Ressource (`Rsp1`), da sich der Content zu dem Zeitpunkt `t2` geändert hat. Diese Version legt

der Web-Cache in seinem Speicher ab und gibt sie gleichzeitig an den Web-Client weiter (`Rsp2`).

3.2 Unterstützung durch HTTP/1.1

Die Spezifikation des HTTP/1.1 festgelegt im **RFC 2616** greift die Ideen der Vorgängerversion auf und integriert sie vollständig. Darüber hinaus werden weitere Zeilen definiert, die zur Aktualitätsprüfung genutzt werden können:

- `ETag` (im Response-Header)
- `If-Range` (im Request-Header)
- `If-Match` (im Request-Header)
- `If-None-Match` (im Request-Header)
- `If-Unmodified-Since` (im Request-Header)
- `Age` (im Response -Header)

Mit der Einführung des `ETag` (=Entity Tag) ist eine Versionskontrolle des Web-Contents möglich. Ein Response-Header kann diese Angabe enthalten und somit dem Empfänger ermöglichen die gespeicherte Version eindeutig zu identifizieren.

Die Headerzeilen `Range` und `If-Range` dienen zur effizienten Anfrage fehlender Dateifragmente. Wird beispielsweise der Download einer großen Datei abgebrochen und wenig später wieder aufgenommen, so ist es dem Webcache möglich dem Ursprungsserver über die Zeile `Range` mitzuteilen, welchen Teil der Datei er bereits lokal verfügbar hat. Über die Zeile `If-Range` kann der Web-Cache nun den fehlenden Teil anfragen unter der Bedingung dass die Datei zwischenzeitlich nicht verändert wurde. Hierzu ist es nötig dem Ursprungswebserver die Version des Web-Contents mitzuteilen. Dies kann in der `If-Range`-Zeile beispielsweise über die Angabe des `ETags` erfolgen oder aber über den Zeitpunkt der letzten Änderung. Diese Information kann der Web-Cache üblicherweise dem zuletzt empfangenen HTTP-Response entnehmen.

Aus Sicht des Ursprungs-Web-Servers wird wie folgt verfahren: Ist die angefragte Teil- Ressource zwischenzeitlich geändert worden, so wird die komplette Ressource erneut übertragen. Anderenfalls liefert er den fehlenden Content-Teil an den Web-Cache aus. In diesem Fall mit HTTP-Response `206 Partial Content`.

Die folgende Grafik stellt den Vorgang noch einmal schematisch dar:

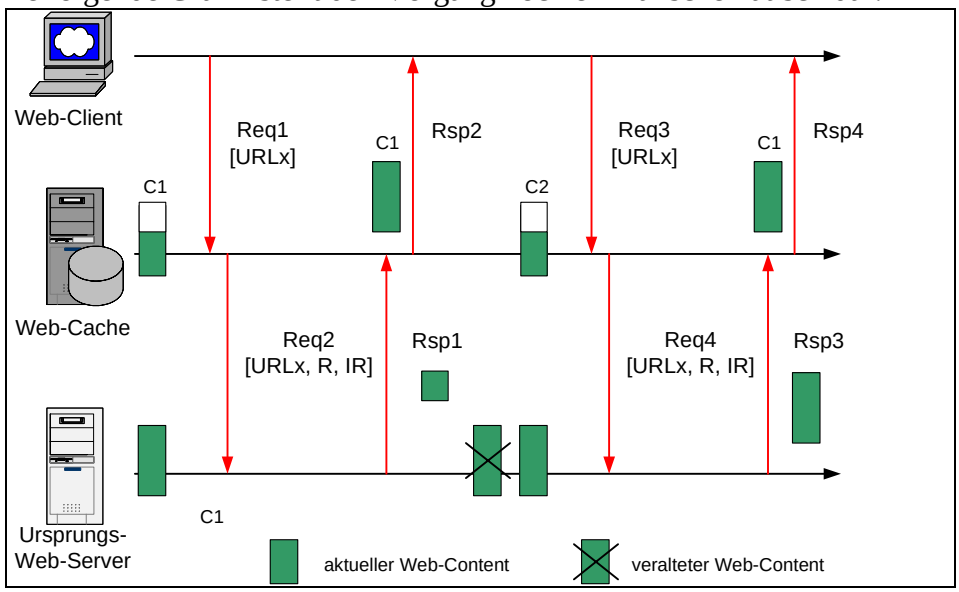

Abb.7: Anwendung von Range/If-Range

Ein Client fragt einen Content `C1` an referenziert durch `URLx` (`Req1`). Der Web-Cache besitzt bereits einen Teil des angeforderten Contents und stellt daher eine Anfrage an den Ursprungs-Web-Server, wobei er den bereits vorhandenen Teil in der Zeile Range angibt und den restlichen Content mit der Zeile `If-Range` anfordert (`Req2`). Da der Content sich zwischenzeitlich nicht geändert hat, antwortet der Ursprungs-Web-Server mit dem fehlenden Teil (`Rsp1`) und der Web-Cache ist nun in der Lage den vollständigen Content `C1` an den Client zu übertragen (`Rsp2`). Der andere Fall wird durch die Anfrage von `C2` dargestellt (`Req3`). Auch hier hat der Web-Client bereits einen Teil des Contents in seinem Cache und fordert wie oben beschrieben den fehlenden Teil an (`Req4`). Da sich der Content zwischenzeitlich geändert hat antwortet hier der Ursprungs-Web-Server jedoch mit dem vollständigen Content von `C2` (`Rsp3`), den der Web-Cache unmittelbar an den Client weitergibt (`Rsp4`).

Wie eingangs erwähnt ist es möglich, dass mehrere Web-Caches verschiedene Versionen eines Web-Contents unter einer bestimmten URL zwischenspeichern. Diese Versionen sind durch die **Entity-Tags** (`ETags`) eindeutig gekennzeichnet. Ein Web-Cache kann unter Angabe eines `ETags` gezielt eine bestimmte Version des Web-Inhaltes anfragen. Dies erfolgt über die Headerzeile `If-match` mit der Angabe einer oder mehrerer `ETag`-Nummeren. Kann der Ursprungs-Web-Server keine dieser Content-Versionen liefern, so antwortet er mit einem HTTP-Response `412 Precondition Failed`.

Auch über die Header-Zeile `If-None-Match` kann ein Web-Cache mehrere `ETags` zur Verifikation an einen Ursprungs-Webserver

übermitteln. Ist eine der angegebenen Versionen gültig wird diese in einem HTTP-Response mit `304 Not Modified` bestätigt. Anderenfalls antwortet der Ursprungswebserver mit `200 OK` und liefert die aktuellste Version des Contents mit.

Über die Zeile `If-Unmodified-Since` kann die Übertragung des Web-Contents davon abhängig gemacht werden, ob sich dieser seit einem bestimmten Zeitpunkt geändert hat. Wenn er unverändert ist, so muss er erneut übertragen werden; ansonsten sendet der Server einen HTTP-Response `412 Precondition Failed`.

Die Header-Zeile `Age` in einem HTTP-Response macht eine Aussage über das Alter des übertragenen Inhalts und beziffert dieses in Sekunden seit dem Zeitpunkt des Abrufens vom Ursprungs-Server. Bei der Zwischenspeicherung in Web-Caches ist somit darauf zu achten diesen Wert entsprechend zu erhöhen bevor der Inhalt an einen anfragenden Client ausgeliefert wird.

3.3 Steuerung des Web-Cache-Verhaltens

HTTP/1.1 ermöglicht das Verhalten eines Web-Caches direkt zu beeinflussen. Sowohl in Anfragen als auch in Antworten kann die Header-Zeile `Cache-Control` genutzt werden, um durch die Angabe von Direktiven zu bestimmen, ob und unter welchen Bedingungen ein Web-Content zwischen gespeichert werden darf.

Diese Zeile hat die allgemeine Form:

```
Cache-Control: <cache-request/response-direktive>
```

Eine HTTP-Nachricht kann mehrere dieser Direktiven enthalten. Die folgende Tabelle zeigt eine Übersicht aller möglichen Funktionen unter Angabe der jeweiligen Verwendung in Response- bzw. Request-Nachrichten.

`no-cache`	**Request**: Der Client benötigt eine aktuelle Version des Contents. **Response**: Die Antwort darf nicht im Cache abgelegt werden.
`no-store`	**Request**: Weder Teil der Anfrage noch des Responses speichern. **Response**: Entspricht `no-cache.`
`max-age`	**Request**: Der Content darf nicht älter als x-Sekunden sein. **Response**: Die Gültigkeit des Contents beträgt x-Sekunden.

`no-transform`	Die Veränderung der übertragenen Web-Inhalte wird unterbunden (z.B. keine zusätzliche JPEG-Komprimierung).
`max-stale`	**Request**: In Verbindung mit max-age akzeptiert der Client auch bereits abgelaufenen Content.
`min-fresh`	**Request**: Gibt die minimale Zeit in Sekunden an, die der Content noch aktuell sein muss um von dem Client akzeptiert zu werden.
`only-if-cached`	**Request**: Die Anfrage darf nur mit gecachten Inhalten beantwortet werden. Es ist keine Aktualitätsprüfung erlaubt.
`public`	**Response**: Das Caching der Antwort ist uneingeschränkt möglich.
`private`	**Response**: die Antwort des Ursprungs-Web-Servers ist ausschließlich für den anfragenden Client bestimmt.
`s-maxage`	**Response**: Anfragen können während der Gültigkeit aus dem Cache beantwortet werden. Danach muss die Aktualität geprüft werden.
`must-revalidate`	**Response**: Durch Angaben in `max-age` bzw. `expires` abgelaufener Content muss revalidiert werden. `Max-stale` wird ggf. ignoriert.
`proxy-revalidate`	**Response**: Entspricht `must-revalidate`. Bezieht sich allerdings nur auf Web-Caches in Web-Proxies.

Für weitere Informationen zu diesen Direktiven wird auf die Ausführungen in `RFC 2116` (Spezifikation des HTTP/1.1 Protokolls) verwiesen.

4 Web-Caching in lokalen Netzwerken

Beim Einsatz von Web-Caching in lokalen Netzwerken ist zwischen zwei grundlegenden Ideen zu unterscheiden:

- **Proxy-Caching-Lösung**
- **Transparente Caching-Lösung**

Proxy-Caching-Lösungen stellen die am häufigsten anzutreffende Form dar. Hier wird der Cache-Server gleichzeitig als **Web-Proxy** genutzt. Transparente Caching-Lösungen hingegen führen den gesamten Internet-Verkehr (HTTP, FTP, E-Mail, ...) über einen Cache-Server.

4.1 Proxy-Caching-Lösung

Bei dieser Lösung dient ein Web-Cache-Server gleichzeitig als **Proxy-Server** für alle Rechner eines lokalen Netzes. Die Clients müssen folglich so konfiguriert werden, dass sie alle HTTP-Requests an den Web-Cache-

Server stellen. Dieser ist über einen Router an das Internet angebunden und greift somit als einziger Rechner direkt auf das Internet zu, wie die folgende Grafik anschaulich zeigt.

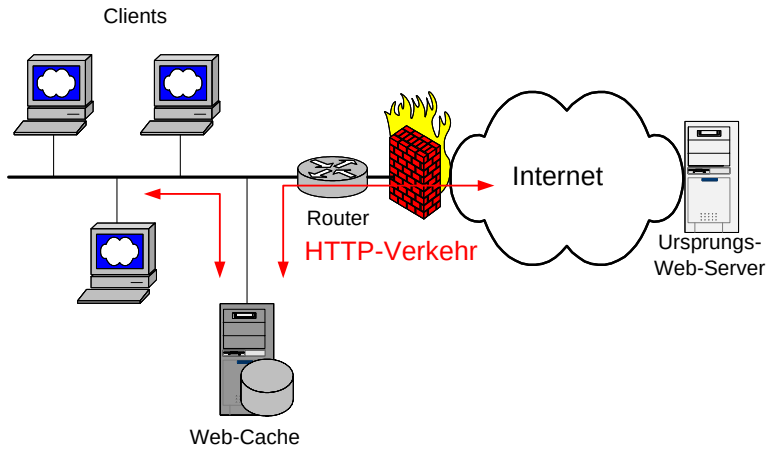

Abb.8: Schema einer Proxy-Caching-Lösung

Vorteile sind bei dieser Lösung eine Bündelung des HTTP-Verkehrs und eine Entlastung des Netzes. Da das Nutzerverhalten der Anwender in einem lokalen Netzwerk zumindest annäherungsweise gleichartig sein sollte erhöht sich die Wahrscheinlichkeit für Cache Hits. Nachteilig ist jedoch die Tatsache, dass jeder Client explizit auf die Verwendung des Proxys eingerichtet werden muss.

Der Konfigurationsaufwand lässt sich jedoch verringern, indem man diese Aufgabe durch die so genannte **Proxy-Automated-URL** (PAC-URL) erledigen lässt. Diese verweist auf eine Datei, die an zentraler Stelle liegt und die Konfigurationsangaben enthält. Änderungen sind somit nur noch an einer Stelle durchzuführen. Der Verweis auf die PAC-URL muss jedoch wie gehabt jedem Web-Browser eingetragen werden (manuell oder durch Skripte).

Einen ähnlichen Ansatz verfolgt **WPAD** (Web Proxy Auto Discovery). WPAD ist ein Protokoll zur automatischen Suche eines Konfigurationsscriptes, welches von Inktomi, Microsoft, Real Networks, and Sun Microsystems erarbeitet wurde, bisher jedoch noch nicht standardisiert ist. Ein WPAD-fähiger Client nutzt **DHCP** (Dynamic Host Configuration Protocol), **SLP** (Service Location Protocol) sowie **DNS** (Domain Name System) um einen lokalen Cache-Service zu ermitteln. Bisher wurde WPAD nur als Internet-Draft veröffentlicht. Seit Version 5.0 (bzw. Version 1.0 Mozilla Firefox) unterstützt der MS-Internet Explorer diese Funktion.

4.2 Transparente Caching-Lösung

Im Falle einer transparenten Caching-Lösung entfällt die Notwendigkeit die Web-Browser konfigurieren zu müssen. Hier wird der gesamte Internetverkehr über den Web-Cache-Server geleitet. Dieser funktioniert als Router und wird von den Clients als Standard-Gateway angesprochen. Diese Konfigurationseinstellung lässt sich also unmittelbar über DHCP realisieren. Für die Clients ist das Vorhandensein eines Cache-Mechanismus vollständig unsichtbar.

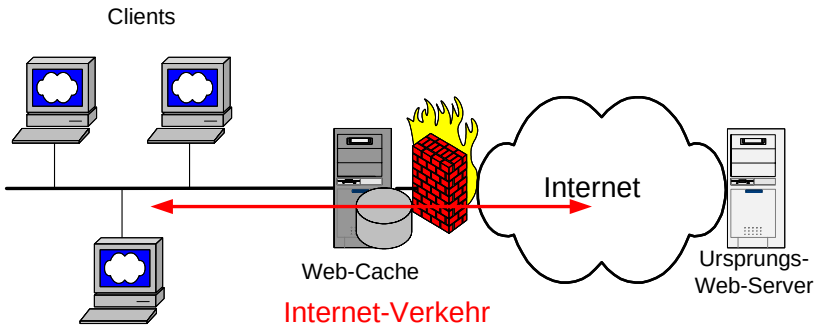

Abb.9: Schema einer transparenten Caching-Lösung

Eine proprietäre Lösung für transparentes Caching hat die Firma Cisco mit dem Protokoll **WCCP** entwickelt (Web Cache Communication Protocol). Dieses Protokoll ermöglicht die transparente Anbindung eines Cache-Servers an einen Router. Sämtliche HTTP-Request von Clients an den Router werden an den Cache-Server weitergeleitet. Kann der Cache-Server die Anfrage beantworten (Cache Hit) so übergibt er den Content an den Router, der ihn an den Client liefert. Bei einem Cache-Miss wird der HTTP-Request an den Ursprungs-Web-Server übermittelt. Der eintreffende Response wird sowohl an den Client als auch an den Cache-Server weitergereicht.

Im Kontext von WCCP ist es auch möglich mehrere Web-Cache-Server zu verwenden. Ab der Version 2 unterstützt WCCP sogar den Einsatz mehrer Router, um die Anfragen an verschiedene Web-Caches zu verteilen.
Somit kann eine Lastverteilung stattfinden und es ist eine gewisse Redundanz im Hinblick auf mögliche Ausfälle gegeben. Auch erhöhten Sicherheitsbedürfnissen wird Rechnung getragen durch Unterstützung einer Authentifizierung zwischen Routern und Cache-Servern nach MD5.

5 Verbundsysteme von Web-Caches

Bei einem Verbundsystem von Web-Caches sind mehrere Cache-Server in einer hierarchischen Struktur miteinander verbunden. Dabei können einem einzelnen Server mehrere sog. Nachbar-Web-Caches zugewiesen werden,

die im selben Netz integriert sind. Die Nachbarn werden weiterhin unterschieden als:

- untergeordnete Kinder (Web-Cache-Children),
- Geschwister auf gleicher Hierarchieebene (Web-Cache-Siblings),
- übergeordnete Eltern (Web-Cache-Parents).

Die Eltern sind in der Regel am Internetzugangsknoten installiert und befinden sich den Ursprungs-Web-Servern am nächsten, wobei die Kinder demnach am weitesten davon entfernt sind.

Im Falle eines Cache-Miss eines Web-Cache-Servers kann dieser zunächst überprüfen, ob das bei ihm nicht vorhandene Web-Objekt von einem seiner Nachbarn beschafft werden kann, um den Zugriff auf den entfernten Ursprungs-Web-Server zu vermeiden. Mit Hilfe von Verbundsystemen lässt sich also die Effizienz des Web-Caching sich erheblich steigern.

5.1 Funktionsweise eines Web-Cache-Verbundsystems

Innerhalb eines Verbundsystems kommunizieren die Cache-Server untereinander über das **ICP-Protokoll**. Bei der Browser-Anfrage eines Web-Objekts (mittels HTTP-Request) an einen **Cache-Verbund** prüft zunächst Cache-Server P (CS P), ob es sich in seinem lokalen Speicher befindet. Wenn dies nicht der Fall ist (Cache-Miss), so befragt er gleichzeitig sämtliche Nachbarn (CS 1-3) mit einem ICP-Request. Wenn er von einem eine positive Antwort (grün dargestellt) erhält (**Neighbor-Hit**), wird das Web-Objekt von diesem Nachbarn angefordert, zwischengespeichert und in einem HTTP-Response an den Browser gesendet. Ansonsten richtet CS P einen HTTP-Request an den Ursprungs-Web-Server.

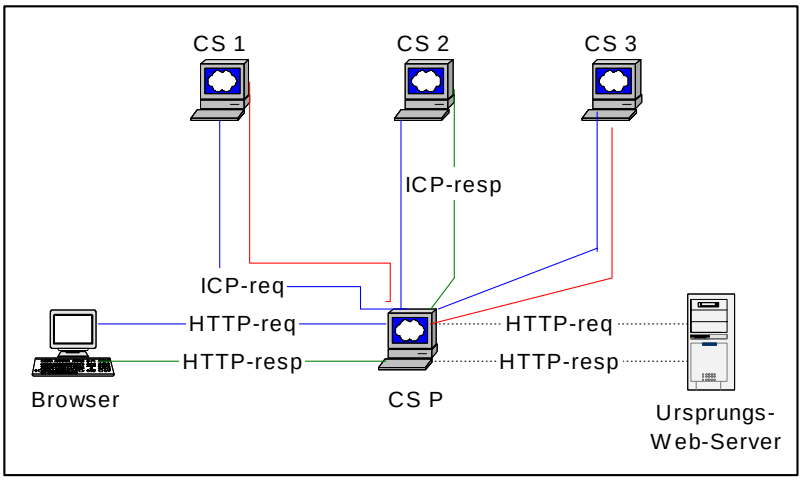

Abb.10: Web-Cache-Verbund mit Proxy

Da sich im worst case, wenn alle Nachbarn mit einem Cache-Miss antworten, die Wartezeit für den Client bzw. Browser erhöhen würde, muss der Austausch der ICP-Nachrichten zwischen den einzelnen Cache-Servern

möglichst schnell erfolgen. In einem Verbundsystem werden deshalb in der Regel ICP-Requests per Mutlicasting versendet. Die Nachbar-Caches werden so konfiguriert, dass ICP-Requests bevorzugt beantworten werden. Zusätzlich erfolgt der Versand der ICP-Nachrichten mittels kleinerer UDP Paketen, da diese im Vergleich zu TCP eine schnellere Antwort ermöglichen. Darüber hinaus unterstützt ICP das Multiplexen von mehreren ICP-Requests über eine persistente TCP-Sitzung, was jedoch in der Praxis selten zur Anwendung kommt.

In einem Verbundsystem mit mehreren Caches wird bei einem Neighbor-Hit oft auf den direkten Versand des Web-Objekts (an den Client) verzichtet, da sonst bei mehreren Hits das Objekt mehrfach geschickt würde. Dies wäre ein ineffizientes Verfahren, da es zum einem die Netzlast erhöhen würde und der Client lediglich die Erste empfangene Nachricht verarbeiten und die restlichen verwerfen würde.

Zudem ergibt sich das Problem, dass der Browser nicht weiß, an welchen Cache-Server er seine Anfrage richten soll. Da er kein ICP unterstützt, wird ein Web-Proxy als Vertreter der Web-Clients eingesetzt (im Beispiel Cache-Server P). Dieser überprüft, ob deren Anfragen von einem der Nachbarn bedient werden können.

5.2 Kooperatives Web-Caching

Beim Einsatz mehrerer, miteinander kooperierender Web-Cache-Server spricht man auch von kooperativem Web-Caching.

Kooperatives Web-Caching kann beispielsweise stattfinden:

- innerhalb eines **Web-Cache-Clusters** (großes Verbundsystem).
- zwischen dem Web-Cache-Server eines Unternehmens und dem eines ISP des Unternehmens,
- zwischen Web-Cache-Servern eines regionalen ISP und Web-Cache-Servern bei einem nationalen ISP.

Die Web-Cache-Server der Zweigstellen eines Unternehmens werden so konfiguriert, dass sie zuerst eine ICP-Anfrage (Query) an den Web-Cache-Server des ISP richten, um zu prüfen, ob dieser das gewünschte Web-Objekt in seinem Speicher hat. Bei einem Cache-Hit sendet der Web-Cache-Server der Zweigstelle einen HTTP-Request an den Web-Cache-Server beim ISP, bei einem Cache-Miss an den Urpsrungs-Web-Server.

6 Das ICP-Protokoll

Web-Caches in einem Verbundsystem kommunizieren in der Regel über das **Internet Cache Protocol** (ICP). Es stammt vom Harvest Project der University of Boulder (Colorado) ab und wurde 1994 während eines

Caching-Projekts beim National Laboratory for Applied Network Research entwickelt. Zur Zeit ist die Version 2 (ICPv2) aktuell. Die gängigste Software zur ICP-Implementierung ist **Squid**. Squid ist für unterschiedliche Unix-/Linux-Derivate sowie für WinNT/Win2k verfügbar.

Merkmale des ICP:

- Protokoll zur Übermittlung von Nachrichten zwischen benachbarten Web-Cache-Servern, die in einer Hierarchie zueinander stehen können.
- Anwendungsprotokoll der Schicht 5, Port 3130. Aus Effizienzgründen erfolgt Übermittlung von ICP-Nachrichten standardmäßig mittels UDP, TCP möglich.
- Unterstützung von Multiplexen mehrerer ICP-Requests mittels TCP.
- Keine Mechanismen zur Validierung des Web-Content.

Die Anfrage nach einem Web-Objekt bei einem Nachbarn geschieht mittels ICP. Das Holen des Objekts erfolgt jedoch per HTTP und da dieses auf TCP aufsetzt, muss vorher eine TCP-Verbindung zum Nachbar-Server aufgebaut werden.

Bei einem Cache-Verbund mit hierarchischem Aufbau werden die Nachbarn beispielsweise konfiguriert als Siblings und Parents. Während sich Siblings innerhalb der gleichen Hierarchiestufe befinden, ist ein Parent ein benachbarter Cache-Server innerhalb der nächsten Hierarchiestufe.

An einen Sibling darf nur dann ein HTTP-Request von einem Nachbarn gesendet werden, wenn dieser das gewünschte Dokument besitzt, also, bei Signalisierung eines Cache-Hit. An einen Parent darf ein HTTP-Request auch dann gesendet werden, wenn er nicht über das Dokument verfügt, also bei Signalisierung eines Cache-Miss.

Es gibt neben ICP verschiedene alternative Ansätze zur Kommunikation im Cache-Verbund-System, z.B. das sog. **Cache-Digest**, das von der IETF spezifizierte **HTCP** Protokoll, oder das **Cache Array Routing Protocol** (CARP) aber darauf soll hier nicht mehr eingegangen werden.

6.1 Konfiguration von ICP

Es folgen zwei Beispiele, in welcher Weise Cache-Server in einem Verbund mittels ICP konfiguriert werden können.

6.1.1 ICP nur zwischen Siblings

ICP wird lediglich lokal verwendet. Alle Web-Cache-Server innerhalb des Unternehmens sind derart vernetzt, dass sie zueinander als benachbarte Siblings gelten. Beim ISP sind ebenfalls alle Web-Cache-Server zueinander

Siblings und benutzen gleichermaßen ICP für Anfragen untereinander. Zwischen einem Web-Cache-Server des Unternehmens und dem Web-Cache-Server beim ISP wird ausschließlich HTTP verwendet. Letzterer ist gegenüber dem Web-Cache-Server des Unternehmens ein Parent.

Mittels ICP-Anfrage im Unternehmen kann bei den Nachbarn ein bestimmtes Objekt angefragt werden. Falls es dort nicht vorhanden ist, erfolgt ein HTTP-Request an einen Web-Cache-Server beim ISP (Parent). Wenn dieser das Objekt auch nicht in seinem Speicher hat, fordert er es entweder von einem seiner Nachbarn oder direkt vom Urpsrungs-Web-Server an. Das Objekt reicht er dann an den Web-Cache-Server im Unternehmen weiter, der es mit HTTP angefordert hatte.

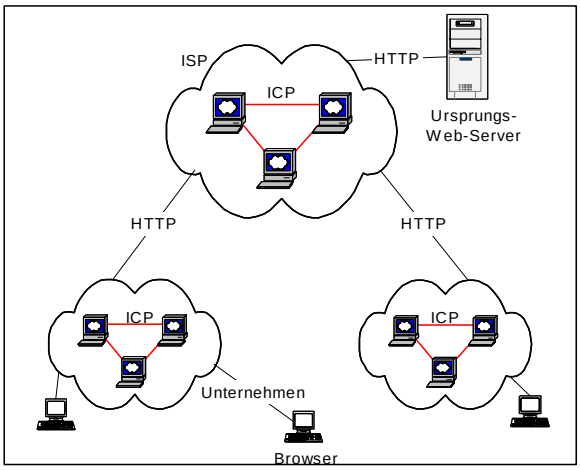

Abb. 11: ICP lokal in einem Unternehmen

6.1.2 ICP zwischen Siblings und Parents

In diesem Beispiel wird ICP zwischen mehreren Standorten eines Unternehmens verwendet wird. Hierbei sind alle Web-Cache-Server innerhalb eines Standorts, sowie innerhalb der Zentrale zueinander Siblings. Die Web-Cache-Server der Zentrale sind gegenüber denen der Niederlassungen Parents.

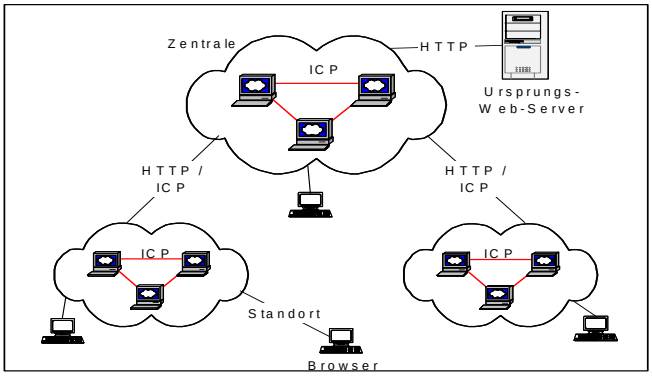

Abb.12: ICP zwischen den Standorten

Falls beispielsweise in Standort 1 ein bestimmtes Objekt angefordert wird, jedoch bei keinem der Nachbarn vorhanden ist, erfolgt zunächst ein HTTP-Request vom Web-Cache-Server des Standorts an einen Web-Cache-Server in der Zentrale und von dort aus an den Ursprungs-Web-Server.

6.2 Funktion von ICP

Abbildung 13 zeigt beispielhaft die Kommunikation des ICP, wie sie bei einem System in Abb. 12 ablaufen würde. ICP wird hier nur zwischen den Siblings eines Unternehmens verwendet.

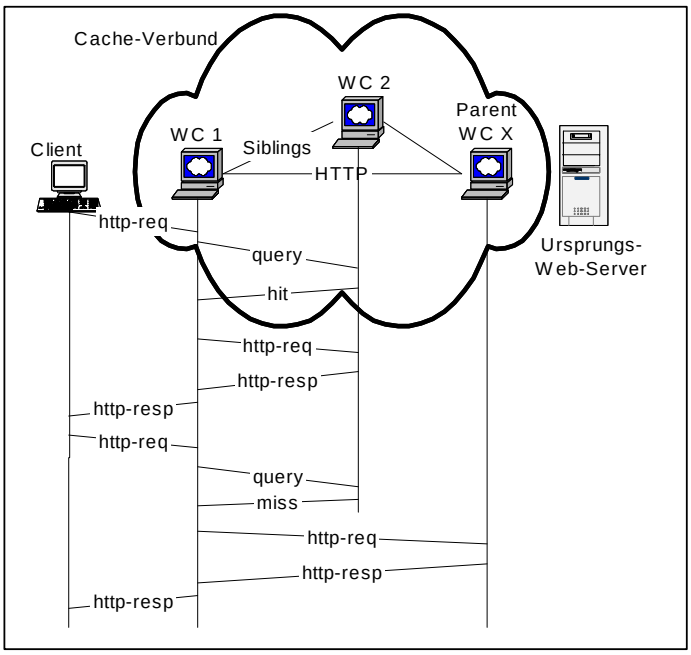

Abb. 13: Beispiel für ICP-Ablauf

Dargestellt sind in Abbildung 13 zwei Situationen:

Situation 1, Hit bei einem Sibling:

1) Ein Browser richtet eine HTTP-Anfrage an Web-Cache-Server 1 (WC 1). Da das Objekt nicht in seinem Speicher vorhanden ist, sendet er eine ICP-Anfrage (ICP_OP_QUERY) an seine Siblings, hier nur Web-Cache-Server 2 (WC 2). Dieser antwortet positiv mit einem ICP_OP_HIT an Web-Cache-Server 1.
2) Web-Cache-Server 1 richtet einen HTTP-Request an Web-Cache-Server 2, um das Objekt von ihm zu holen (d.h.nach vorherigem TCP Aufbau).
3) Web-Cache-Server 2 sendet das Objekt mit einem HTTP-Response an Web-Cache-Server 1, welcher es bei sich speichert.
4) Web-Cache-Server 1 sendet das Objekt an den Browser.

Situation 2, Miss bei allen Siblings:

1) Ein Client richtet eine HTTP-Anfrage an Web-Cache-Server 1. Da das Objekt nicht in seinem Speicher vorhanden ist, sendet er eine ICP-Anfrage (ICP_OP_QUERY) an Web-Cache-Server 2. Dieser antwortet negativ mit einem ICP_OP_MISS an Web-Cache-Server 1.
2) Web-Cache-Server 1 richtet HTTP-Request an Web-Cache-Server X (WC X), seinen Parent, um das Objekt von ihm zu holen.
3) Web-Cache-Server X hat es im Speicher und sendet es in einem HTTP-Response an Web-Cache-Server 1, welcher es bei sich speichert. Falls das Objekt nicht im Speicher des Parent-Servers vorhanden ist, holt dieser es vom Ursprungswebserver.
4) Web-Cache-Server 1 sendet Objekt an den Browser (Client).

7 Zusammenfassung und Ausblick

Web-Caching ist ein effizientes Mittel zur Reduzierung von Netzlast und Downloadzeiten und somit zur Senkung der Onlinekosten. Wichtige Aspekte sind dabei die Content-Validierung und Kommunikation zwischen den verschiedenen Cache-Servern.

Als ein komplexes Thema ist Web-Caching seit mehreren Jahren Gegenstand umfangreicher Forschungen und Entwicklungen. Aktuell werden u.a. Algorithmen zur Unterstützung des sog. Prefetching entwickelt. Web-Caches, die prefetching-fähig sind, können den Bedarf an bestimmten Web-Objekten voraussehen und diese anfordern, bevor HTTP-Anfragen danach eintreffen. Ein weiteres Thema ist z.B. das Adaptive Web-Caching, wobei Prinzipien und Protokolle entwickelt werden, um verschiedene Web-Caches mit ähnlichen Speicherinhalten zu einem Verbund zusammenzufassen. Als Protokolle kommen dabei das Cache Group Management Protocol (CGMP), sowie das Content Routing Protocol (CRP) zum Einsatz.

Das Internet wird in Zukunft als Plattform für multimediale Dienste und Kommunikation weiter ausgebaut. Sinnvolle Einsatzgebiete für Webcaches sind z.B. Plattformen für den Musicdownload, oder die sog. Content Delivery Networks (Streaming-Media, E-Learning, Video on Demand etc.). Da Web-Cache-Systeme eine grundlegende Basis für solche Dienste bilden, werden sie auch weiterhin von großer Bedeutung sein.

8 Quellenverzeichnis

[BAD] Anatol Badach, *Web-Technologien*, Hanser-Verlag, München, 2003
[DAV] Brian D. Davison, *Web Caching and Content Delivery Resources*, http://www.web-caching.com
[PET] Larry L. Peterson/Bruce S. Davie, Computernetze, dpunkt-Verlag, Heidelberg, 2004
[RFCa] RFC 1945 - *HTTP/1.0.*, http://www.ietf.org/rfc/rfc1945.txt
[RFCb] RFC 2616 – *HTTP/1.1*, http://www.ietf.org/rfc/rfc2616.txt.
[WIK] WIKI, *Web Caching Wiki*, http://wiki.web-cache.com/cgi-bin/moin
[WIP] Online Enzyklopädie, *Online Enzyklopädie*, http://www.wikipedia.org
[WÖH] Heiko Wöhr. *Web-Technologien*. dpunkt-Verlag, Heidelberg, 2004

Alle Hyperlinks auf Funktion gestestet am 04.12.2004.