Der Webproxy Squid

Ein Überblick und ein wenig (viel) HTTP

Dirk Geschke

Linux User Group Erding

28. April 2010

Gliederung

HTTP

Überblick

- Hypertext Transfer Protocol dient der Übertragung von Daten
- in erster Linie Webseiten aka HTML-Seiten
- realisiert das World Wide Web
- verwendet TCP, in erster Linie Port 80
- unterstützt verschiedene Methoden

RFC-Standards

- HTTP 0.9: RFC-1945
- HTTP 1.0: RFC-1945
- HTTP 1.1: RFC-2616
- Authentisierungen: RFC-2617 - basic und digest
- HTTP over TLS: RFC-2818
- Erweiterungen wie WEBDAV: RFC-4918

Technisches Konzept: Request

- Requests bestehen aus
  Methode `Leerzeichen` URL `Leerzeichen` Protocol
- gefolgt von Headerfeldern
- gefolgt von einer Leerzeile
- gefolgt von optionalen Daten im Body, abhängig von der Methode
- der Header besteht aus ASCII-Zeichen

Uniform Resource Locator

Beispiel:

`http://User:Pass@www.google.de:80/search?q=lug-erding&...`

- Protokoll: http, https, ftp, file, ...
- Benutzername: optional
- Passwort: optional
- hostname: Zielsystem
- Port: optionale Portnummer, default ist 80
- URL-Pfad Angabe der Daten auf dem Zielsystem, auch relative URL
- Query-String: optionale Daten für CGI-Programme

Zeichenbeschränkungen

- nicht erlaubte Zeichen sind:

  `% / . .. # ? ; : $ , + @ & = 0-0x1f 0x7f-`

- unsichere Zeichen sind:

  `{} [ ] ~ | \ < > '`

- Lösung: Kodierung mittels %{HEX}, z.B. Leerzeichen: %20
- Tilde wird häufig für persönliche Webseiten verwendet:
  `http://www.example.com/~geschke`

Methoden

GET	Holen der Daten
HEAD	Anforderung des Headers
POST	Senden von Daten im Body an den Server
PUT	Ablegen von Daten / Ressourcen
OPTIONS	Abfrage der Fähigkeiten des Servers
DELETE	Löschen von Ressourcen
LINK	Erstellen von Links
UNLINK	Löschen von Links
PATCH	Patchen von Dateien
TRACE	Verfolgen des Requests
CONNECT	Proxy-Methode für SSL-Verbindungen

Technisches Konzept: Response

- Antworten bestehen aus
  Protokoll `Leerzeichen` Statuscode `Leerzeichen` Text
- Headerfelder, gefolgt von Doppelpunkt und Werten
- Leerzeile
- Body auch Entity Body genannt, abhängig von der Methode
- bei neueren Protokollen sind persistente Verbindungen möglich

Statuscodes

- 1xx informative Nachrichten, definierter Bereich 100-101
- 2xx erfolgreiche Aktion, definierter Bereich 200-206
- 3xx redirects, definierter Bereich 300-305
- 4xx Fehler vom Client, definierter Bereich 400-415
- 5xx Fehler vom Server, definierter Bereich 500-505

Statuscodes

Code	Bedeutung
100	Continue
101	Switching Protocols
200	OK
201	Created
202	Accepted
203	Non-Authoritive Information
204	No Content
205	Reset Content
206	Partial Content

Statuscodes

Code	Bedeutung
300	Multiple Choices
301	Moved Permanently
302	Found
303	See Other
304	Not Modified
305	Use Proxy
306	(Unused)
307	Temporary Redirect

Statuscodes

Code	Bedeutung
400	Bad Request
401	Unauthorized
402	Payment Required
403	Forbidden
404	Not Found
405	Method not allowed
406	Not Acceptable
407	Proxy Authentication Required
408	Request Timeout
409	Conflict

Statuscodes

Code	Bedeutung
410	Gone
411	Length Required
412	Precondition Failed
413	Request Entity Too Large
414	Request URI Too Long
415	Unsupported Media Type
416	Requested Range Not Satisfiable
417	Expectation Failed

Statuscodes

Code	Bedeutung
500	Internal Server Error
501	Not Implemented
502	Bad Gateway
503	Service Unavailable
504	Gateway Timeout
505	HTTP-Version Not Supported

Authentisierungen

basic Benutzername und Passwort werden unverschlüsselt aber Base64-kodiert im HTTP-Header übertragen

digest Hash (MD5) aus Passwort, *Realm* und einer *Nonce* wird im HTTP-Header übertragen
$\Longrightarrow$ effektiver Schutz gegen mitsniffen.

NTLM NT LAN Manager

- kein offizieller Standard
- proprietär
- gegenüber Webserver im Internet schwierig.
- erhöhter Traffic
- keine Passwortabfrage, läuft im Hintergrund

Headerfelder - allgemeine

Connection Anleitungen für die Verbindung wie `keep-alive` oder `close`, eigentlich zu entfernende Headerfelder für Proxys

Date Zeit, wann der Request bearbeitet wurde

Via Angabe über welchen Proxy die Anfrage lief

Cache-Control Einflussmöglichkeiten auf den Cache, z.B. `no-cache`, `no-store`

Pragma ähnlich zu `Cache-Control`, *deprecated*

Headerfelder - Request

Host	Hostname und Port des Servers, Pflicht bei HTTP/1.1!
From	E-Mailadresse des Surfers!
Referer	Wo war der Client vorher?
User-Agent	Welcher Browser wird verwendet?

Accept-Headerfelder - Request

Präferierungen über *quality values* aka *q*-Werte möglich

Accept Welche Media-Typen werden angenommen?

Accept-Charset Welche Zeichensätze sind gewünscht?

Accept-Encoding gzip, compress

Accept-Language Welche Sprachen sind okay?

bedingte Headerfelder - Request

Einfluss auf Server: Alle Daten oder nur den Header?

If-Match bestimmtes (altes) Dokument

If-Modified-Since habe gecachte Version von

If-None-Match für ETags

If-Range Ist der Bereich noch korrekt?

If-Unmodified-Since altes Dokument

Range Anforderung von einem bestimmten Bereich der Daten, z.B. Bytes 500-1000

Headerfelder - Request

Felder für die Authorisierung von Clients und für Proxys

Authorization Verfahren und Wert

Cookies Keks vom Server

Cookies2 Version des Kekses

Max-Forwards maximale Zahl der Proxys $\Longrightarrow$ für TRACE-Methode

Proxy-Authorization explizites Feld für Proxy

Proxy-Connection explizites Feld für Proxy

Headerfelder - Response

Age Alter des Ergebnisses, Proxys müssen dies angeben

Server Name und Version des Webservers, z.B. Apache/2.2.14 (FreeBSD)

Vary es ist nur eine Variante basierend auf Accept-Header

Proxy-Authenticate Liste der Authensierungsanforderungen vom Proxy

Set-Cookie überträgt einen Keks

Set-Cookie2 anderes Keksformat

WWW-Authenticate Liste der Authentisierungsanforderungen vom Server

Headerfelder - Daten

Location Wo liegen die Daten wirklich $\Longrightarrow$ *Redirect*

Content-Type Welcher Art sind die Daten?

Content-Encoding Sind die Daten gepackt?

Content-Length Anzahl der Bytes der Daten $\Longrightarrow$ *persistente Verbindungen*!

Content-Language Sprache

Content-Range Bereich der Daten

Content-MD5 MD5-Summe über die Daten

Headerfelder für Caches

ETag *Entity Tag*, ändern sich die Daten, so ändert sich der ETag $\Longrightarrow$ starker Hinweis auf Gültigkeit von gecachten Daten

Expires Bis wann ist der Inhalt gültig?

Last-Modified Wann wurde der Inhalt das letzte Mal geändert?

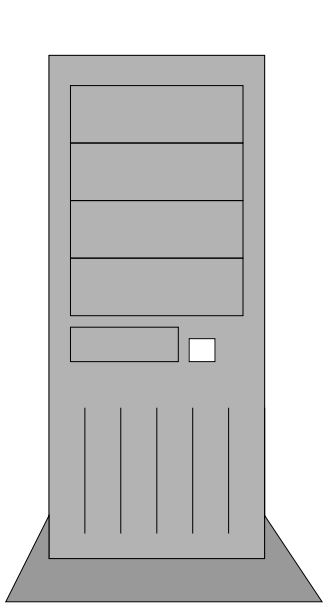
Performance-Optimierungen

multiple Verbindungen paralleler Download mehrerer URLs $\Longrightarrow$ mehrere TCP-Verbindungen, Variante: *Ranges*!

persistente Verbindungen bestehende Verbindung mehrfach benutzen $\Longrightarrow$ nur eine TCP-Verbindung, nur ein TCP-Handshake

Pipelining mehrere Requests in Folge in einer Verbindung absetzen $\Longrightarrow$ am schnellsten, aber problematisch!

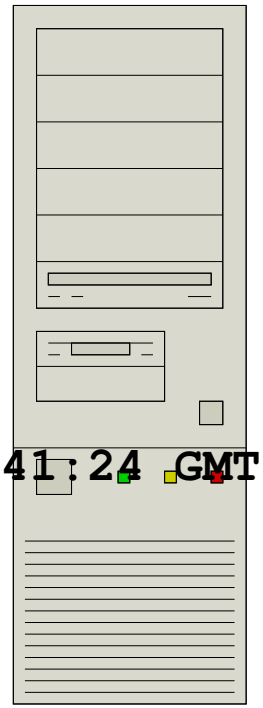
Beispiel



Weblclient

```
HEAD / HTTP/1.0
```

```
HTTP/1.1 200 OK
Date: Tue, 27 Apr 2010 20:14:11 GMT
Server: Apache
Last-Modified: Tue, 13 Apr 2010 12:41:24 GMT
ETag: "1041200f-2bb3-944f8500"
Accept-Ranges: bytes
Content-Length: 11187
Connection: close
Content-Type: text/html
```



Webserver

Webproxys / Squid

Eigenschaften

Proxy	zentraler Drehpunkt
Cache	Einsparung von Internetressourcen
Logdateien	erleichtert die Fehlersuche
ACL	extrem gute Filtermöglichkeiten
Privacy	Alle Anfragen kommen vom Proxy, Headerfelder können gefiltert/modifiziert werden, siehe z.B. `http://panopticlick.eff.org/`
Kommunikation	Client spricht HTTP mit Proxy, auch bei FTP!
Dokumentation	hervorragend!

besondere Eigenschaften

- Bandbreitenmanagement via cache-pools
- Virenscanning via ICAP möglich
- externe Programme einbindbar wie z.B. squidGuard
- Viele Authentisierungsverfahren: Basic, Digest, NTLM, Kerberos, ...
- reverse Proxy möglich
- SSL Bump: Filterung in SSL-Seiten per *squid-in-the-middle*
- IPv6-Support
- QoS-Support
- kaskadierbar: Parents, Siblings, etc.

Besonderheiten bei Proxys/Caches

- Requests mit vollständiger URL
- FTP wird zum Client über HTTP abgewickelt
- CONNECT-Methode zum Tunneln von z.B. `https`
- Methode PURGE zum expliziten Löschen einzelner Requests aus dem Cache (via `squidclient`)
- Prefetching möglich: Vorabfüllung des Caches
- Cachemgr: Auslesen der Squid-Statistiken
- Cluster: Hierarchische Proxys mit diversen Protokollen

Filtermöglichkeiten

- Methode: GET, POST, CONNECT, ...
- Protokoll: http, ftp, ...
- Browser
- URL / reguläre Ausdrücke
- IPs / Domains / Ports / MAC-Adressen (ARP)
- Zeit: Tage, Stunden
- MIME-Types, z.B. Audio
- Erweiterung durch externe ACLs möglich
- Entfernen / Ändern von Headerfeldern
- Auslagerung in Dateien, z.B. URL-Filterlisten

Client-Konfiguration

manuell z.B.: `http_proxy=http://myproxy:3128/`

PAC-Datei Proxy Auto-Configuration via URL, z.B.: `http://myserver/proxy.pac`

- JavaScript-Datei zum Finden von Proxys oder direktem Zugriff

WPAD Web Proxy Auto-Discovery: auomatisches Finden

- PAC-Datei wird von einem Server geladen
- URL via DHCP, option code 252
- SLP, Service Location Protocol
- DNS, `http://wpad.`*Domain*`/wpad.dat`
- DNS, SRV-Records
- DNS, URL aus TXT-Record

Was darf im Cache landen oder nicht?

- POST-Requests: nicht cachen
- Authentisierung erforderlich: nicht cachen
- basierend auf Statuscode z.B. 200, 206, 300, 301, 410 cachebar
- dynamische Seiten
- Verwendung von `Expires` mit aktuellem Datum oder invaliden Eintrag wie `Expires: 0`
- Verwendung von `Pragma: no-cache`
- Cookies, siehe `Cache-Control`
- mit Hilfe des Headerfeldes `Cache-Control`

Cache-Control Werte

private	Nur der eigene Browser darf cachen
public	Proxys dürfen auch cachen, auch bei Authentisierung
no-cache	darf im Cache sein aber Validierung erforderlich, Felder können angegeben werden, die nicht gespeichert werden dürfen, z.B. `no-cache=Set-Cookie`
max-age	Verfallzeit, impliziert `public`
s-max-age	`max-age` für Proxys (*shared cache*), impliziert **kein** `public`
must-revalidate	Aktualität muss überprüft werden
proxy-revalidate	Aktualität muss von Proxys überprüft werden
no-store	darf nicht gecacht werden!

Refresh erzwingen

Firefox & Co SHIFT+Reload

IE CTRL+Reload

$\Longrightarrow$ `If-Modified-Since`-Header wird entfernt

- Reload-Problem: Wenn der Transfer zu lange dauert...

Debugging von Webproxys und Webservern

Logdateien geben oft schon Aufschluss über das Problem, Statutscodes, etc.

telnet einfaches Testen von Servern/Proxys

tcpdump mitsniffen des tatsächlichen Traffics, Tipp: `strings`!

Live HTTP Headers Plugin für Firefox, zeigt die gesendeten und empfangenen Header an.

RFCs Was sagt der Standard?

Praxis

Real-Live-Tests

- Client versus Server
- GET versus POST
- Authentisierung gegen Server
- Authentisierung gegen Proxy
- Client versus Proxy versus Server
- Logdateien
- Problematik von CONNECT
- Spielereien...

Fine!