Hypertext Transfer Protocol

Ingo Blechschmidt
<iblech@web.de>

LUGA

6. Juli 2005

Inhalt

Geschichte

- Erster RFC für HTTP/1.0: RFC 1945, Mai 1996
- Erster RFC für HTTP/1.1: RFC 2616, Juni 1999
- Viele weitere RFC bis jetzt

Verwendung von HTTP

- World Wide Web (inkl. Web Services)
- Aufbauend:
  WebDAV (WWW Distributed Authoring and Versioning),
  Subversion (Versionskontrollsystem)
- Ähnlich: SIP (Session Initiation Protocol) u.a. für VoIP

Typischer Ablauf

- Wunsch: `http://www.pro-linux.de/berichte/`
- Also Verbindung zu `www.pro-linux.de:80` per TCP:

```
GET /berichte/ HTTP/1.1
Host: www.pro-linux.de
Connection: close

HTTP/1.1 200 OK
Date: Mon, 04 Jul 2005 18:17:33 GMT
Server: Apache
Last-Modified: Sun, 10 Apr 2005 21:20:54 GMT
Content-Length: 10883
Content-Type: text/html

<!DOCTYPE html[...]
```

Typischer Ablauf

- Wunsch: `http://www.pro-linux.de/berichte/`
- Also Verbindung zu `www.pro-linux.de:80` per TCP:

```
GET /berichte/ HTTP/1.1
Host: www.pro-linux.de
Connection: close

HTTP/1.1 200 OK
Date: Mon, 04 Jul 2005 18:17:33 GMT
Server: Apache
Last-Modified: Sun, 10 Apr 2005 21:20:54 GMT
Content-Length: 10883
Content-Type: text/html

<!DOCTYPE html[...]
```

Typischer Ablauf

- Wunsch: `http://www.pro-linux.de/berichte/`
- Also Verbindung zu `www.pro-linux.de:80` per TCP:

```
GET /berichte/ HTTP/1.1
Host: www.pro-linux.de
Connection: close

HTTP/1.1 200 OK
Date: Mon, 04 Jul 2005 18:17:33 GMT
Server: Apache
Last-Modified: Sun, 10 Apr 2005 21:20:54 GMT
Content-Length: 10883
Content-Type: text/html

<!DOCTYPE html[...]
```

Typischer Ablauf

- Wunsch: `http://www.pro-linux.de/berichte/`
- Also Verbindung zu `www.pro-linux.de:80` per TCP:

```
GET /berichte/ HTTP/1.1
Host: www.pro-linux.de
Connection: close

HTTP/1.1 200 OK
Date: Mon, 04 Jul 2005 18:17:33 GMT
Server: Apache
Last-Modified: Sun, 10 Apr 2005 21:20:54 GMT
Content-Length: 10883
Content-Type: text/html

<!DOCTYPE html[...]
```

Request-Methoden

- Request-Methode als erste Zeile des Requests
- Generell: *`Methode Pfad`* `HTTP/`*`1.1`*

Verfügbare Methoden

`GET`:	Normales Herunterladen einer Seite
`HEAD`:	„Ich möchte bitte nur die Header.“
`POST`:	Schicken von Daten an den Server (z.B. Formulare)
`TRACE`:	Zurücksenden des gesamten Requests (Debugging!)
`CONNECT`:	Direkte Weiterleitung auf TCP-Ebene (SSL, Proxyketten)

Header-Felder

- Format ähnlich wie bei RFC 822-konformen Mails
- Header beginnend mit `X-`: nicht standardisiert

Oft verwendete Header

`Host:`	Direkte Adressierung (Virtual Hosts!)
`Date:`	Datum der Antwort (nützlich für z.B. Zeitsynchronisation)
`Last-Modified:`	Datum der letzten Änderung
`Server:`	Server-Software (z.B. Apache)
`User-Agent:`	Client-Software (z.B. Lynx, ELinks)
`Content-Length:`	Länge in Bytes des Antwort-Dokuments
`Content-Type:`	MIME-Typ des Antwort-Dokuments (z.B. `text/html`, `application/xml`)

Header-Felder

- Format ähnlich wie bei RFC 822-konformen Mails
- Header beginnend mit `X-`: nicht standardisiert

Oft verwendete Header

`Host:`	Direkte Adressierung (Virtual Hosts!)
`Date:`	Datum der Antwort (nützlich für z.B. Zeitsynchronisation)
`Last-Modified:`	Datum der letzten Änderung
`Server:`	Server-Software (z.B. Apache)
`User-Agent:`	Client-Software (z.B. Lynx, ELinks)
`Content-Length:`	Länge in Bytes des Antwort-Dokuments
`Content-Type:`	MIME-Typ des Antwort-Dokuments (z.B. `text/html`, `application/xml`)

Header-Felder

- Format ähnlich wie bei RFC 822-konformen Mails
- Header beginnend mit `X-`: nicht standardisiert

Oft verwendete Header

`Host:`	Direkte Adressierung (Virtual Hosts!)
`Date:`	Datum der Antwort (nützlich für z.B. Zeitsynchronisation)
`Last-Modified:`	Datum der letzten Änderung
`Server:`	Server-Software (z.B. Apache)
`User-Agent:`	Client-Software (z.B. Lynx, ELinks)
`Content-Length:`	Länge in Bytes des Antwort-Dokuments
`Content-Type:`	MIME-Typ des Antwort-Dokuments (z.B. `text/html`, `application/xml`)

Keep-Alive

- Früher, bei HTTP/1.0:
  Eine TCP-Verbindung für jeden Request
- Nachteil: Ständiges Öffnen und Schließen von TCP-Verbindungen ineffizient
- Daher, seit HTTP/1.1:
  Keep-Alive – Offenhalten der Verbindungen
- Neuer Header: `Connection: close`

Nutzen von Proxies

- Üblicherweise: Client ↔ Server
- Mit Proxies: Client ↔ Proxy ↔ Server

- Traffic- und Zeiteinsparungen (Cache!)
- Möglichkeit der detaillierten Zugangsbeschränkung
- Ausfiltern von invaliden und möglicherweise die Sicherheit kompromittierenden HTTP-Requests
- (Aber: Möglicherweise Sicherheitslücken auch in den Proxy-Daemonen)
- U.U. Erhöhung der Anonymität

Proxies bei HTTP

```
GET http://www.pro-linux.de/ HTTP/1.1
Host: www.pro-linux.de

HTTP/1.1 200 OK
[...]
```

- Kurz: Statt dem Pfad die vollständige URL im Request
- Andere Header wie üblich
- Übliche Header-Ergänzungen durch Proxies:
  `Via`, `X-Forwarded-For`

CONNECT-Methode

```
CONNECT www.pro-linux.de:80 HTTP/1.1
Host: www.pro-linux.de

HTTP/1.1 200 Connection established

GET / HTTP/1.1
Host: www.pro-linux.de

HTTP/1.1 200 OK
[...]
```

CONNECT-Methode

```
CONNECT www.pro-linux.de:80 HTTP/1.1
Host: www.pro-linux.de

HTTP/1.1 200 Connection established

GET / HTTP/1.1
Host: www.pro-linux.de

HTTP/1.1 200 OK
[...]
```

CONNECT-Methode

```
CONNECT www.pro-linux.de:80 HTTP/1.1
Host: www.pro-linux.de

HTTP/1.1 200 Connection established

GET / HTTP/1.1
Host: www.pro-linux.de

HTTP/1.1 200 OK
[...]
```

CONNECT-Methode

```
CONNECT proxy:3128 HTTP/1.1
Host: proxy:3128

HTTP/1.1 200 Connection established

GET http://www.pro-linux.de/ HTTP/1.1
Host: www.pro-linux.de

HTTP/1.1 200 OK
[...]
```

CONNECT-Methode

```
CONNECT proxy1:3128 HTTP/1.1
Host: proxy1:3128

HTTP/1.1 200 Connection established

CONNECT proxy2:3128 HTTP/1.1
Host: proxy2:3128

HTTP/1.1 200 Connection established

GET http://www.pro-linux.de/ HTTP/1.1
Host: www.pro-linux.de

HTTP/1.1 200 OK
[...]
```

Funktioniert das denn wirklich?

- Ja. :)
- `$ ./proxychain --listen=8000 -- \`
  `proxy1:3128 proxy2:3128 [...]`
- Im Browser: Setzen der Proxy auf `localhost:8000`
- `http://whatismyip.com/`,
  `http://www.augustakom.de/speedtest/`
- Meistens sogar Schleifen möglich:
  `$ ./proxychain --listen=8000 -- \`
  `p1:3128 p2:3128 p3:3128 p1:3128 [...]`

Funktioniert das denn wirklich?

- Ja. :)
- `$ ./proxychain --listen=8000 -- \`
  `proxy1:3128 proxy2:3128 [...]`
- Im Browser: Setzen der Proxy auf `localhost:8000`
- `http://whatismyip.com/`,
  `http://www.augustakom.de/speedtest/`
- Meistens sogar Schleifen möglich:
  `$ ./proxychain --listen=8000 -- \`
  `p1:3128 p2:3128 p3:3128 p1:3128 [...]`

Siehe auch

- RFC 1945: Hypertext Transfer Protocol – HTTP/1.0
- RFC 2616: Hypertext Transfer Protocol – HTTP/1.1
- `http://freshmeat.net/projects/theguide/`: Hitchhiker's Guide to the Internet
- `http://m19s28.vlinux.de/iblech/proxychain-luga-20050706.pl`: Proof-of-Concept-Implementierung

Fragen?