## 1.2 HTML/HTTP

### 1.2.1 Kurzüberblick: http im Netzwerk

Zur Verwirklichung der Hypertextidee brauchte man

a) HTML - ein Dateiformat, welches maschinenlesbare Verweise (*links*) enthält,

b) einen (netzwerkfähigen) Leser (*browser*) für das Dateiformat,

c) einen (netzwerkfähigen) Anbieter (*server*) für diese Dateien und

d) HTTP - eine gemeinsame Sprache (Protokoll), in der sich Browser und Server miteinander verständigen können.

Das Protokoll ist einfach gehalten und sowohl vom Aufbau als auch vom Zeichensatz her für Menschen gut lesbar. HTTP benutzt, da es zur TCP/IP-Familie gehört, andere Teile von TCP/IP mit. In Abbildung 1 erkennt man den Schichtaufbau von TCP/IP. Zwei Computer `System 1` und `System 2` sind miteinander per Netzwerkhardware verbunden. Nur die Hardware der beiden

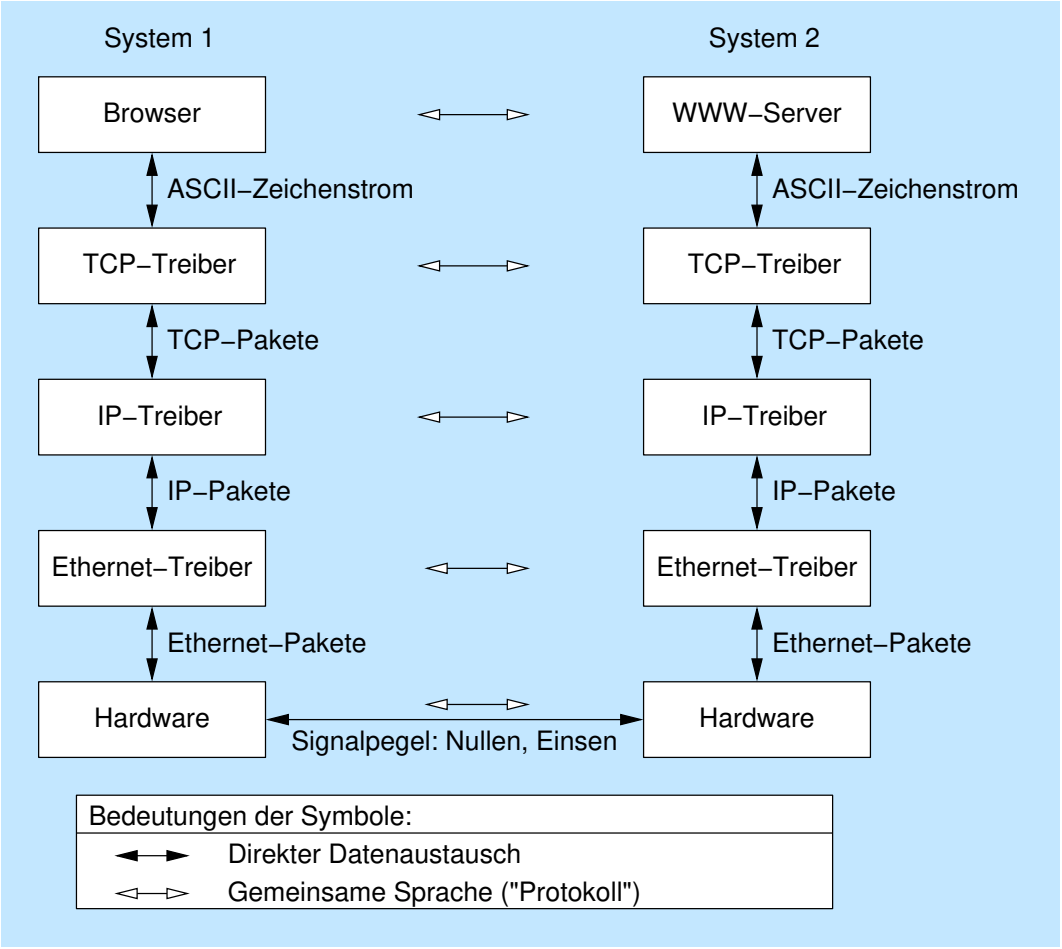


Abbildung 1: Blockbild

Systeme kommuniziert direkt miteinander. Innerhalb eines Systems werden die Daten von einer Schicht an die andere weitergegeben. Von oben nach unten werden sie nach bestimmten Regeln eingepackt, von unten nach oben werden sie auf dem jeweils anderen System wieder ausgepackt.

Damit Ein- und Auspacken bei Empfänger und Sender jeweils gleich laufen (und nicht voneinander abweichen), müssen die gleichen Schichten beider Systeme die gleiche Sprache sprechen.

Die unter HTTP liegenden Schichten werden durch das Netzwerkbetriebssystem (Linux, FreeBSD, Wind.) zur Verfügung gestellt. Für HTTP selbst dagegen sind der Webbrowser und der Webserver zuständig. Damit vereinfacht sich das Blockbild wesentlich (Abbildung 2). TCP/IP stammt

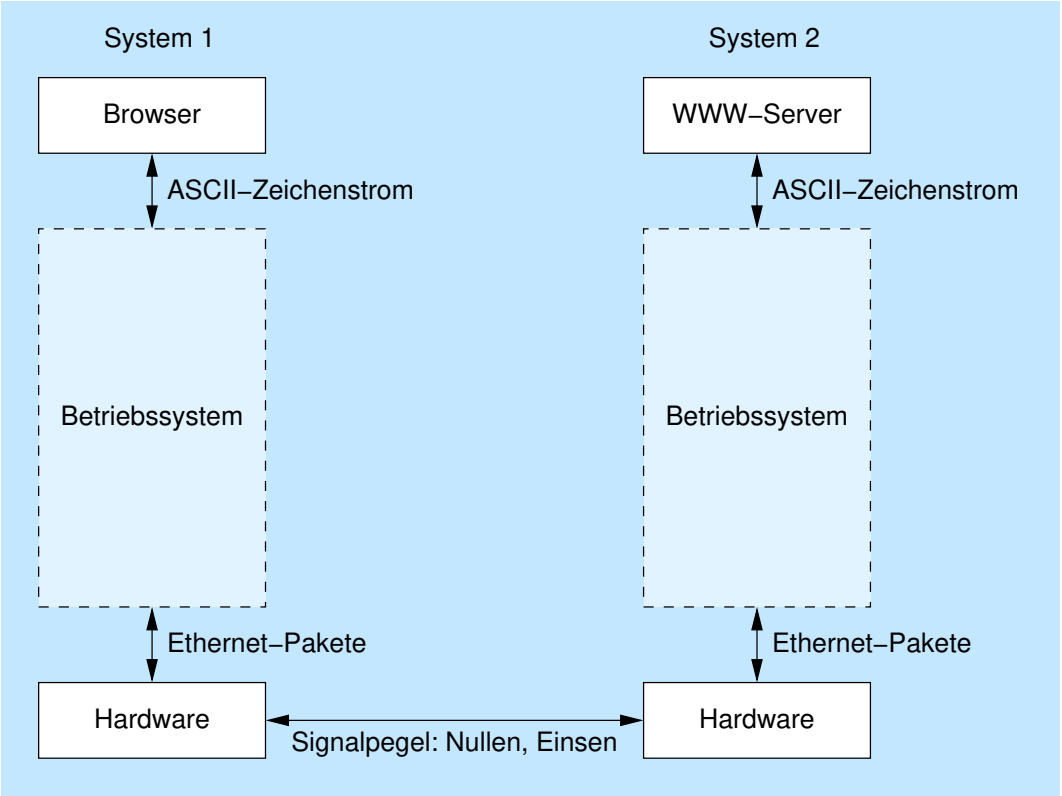


Abbildung 2: Vereinfachtes Blockbild

aus der Welt der Großrechner. Dort ist es üblich, dass auf einem System mehrere Server-Dienste (HTTP, FTP, SMTP) gleichzeitig ansprechbar sind. Wie hat man das nun erreicht? Die Lösung zeigt Abbildung 3.

Jedes System besitzt zur eindeutigen Identifizierung mindestens eine IP-Adresse (bzw. einen Hostnamen). Innerhalb des Systems können (bei IP-Version 4) 65536 Unteradressen unterschieden werden, so genannte Portnummern.

Jedes Serverprogramm kann sich nun eine von 65536 freien Portnummern aussuchen.[1] Zu jedem TCP/IP-Protokoll (der obersten Schicht) existiert eine Standard-Portnummer. Die meisten Server und Clients benutzen, falls nicht ausdrücklich anders angegeben, genau diese Portnummer.

Diese Zuordnung ist auf Linux-Systemen in der Datei `/etc/services` einsehbar. Hier ein Ausschnitt aus dieser Datei:

```
##############################################################################
# Service       port/transp.layer
##############################################################################
ftp-data        20/tcp                  # File Transfer [Default Data]
ftp             21/tcp                  # File Transfer [Control]
ssh             22/tcp                  # SSH Remote Login Protocol
```

[1] Meist werden die Nummern 1-1023 für ankommende Verbindungen (also für Serverprogramme), der Rest für abgehende Verbindungen (also für Clients) reserviert (davon gibt es aber viele Ausnahmen).

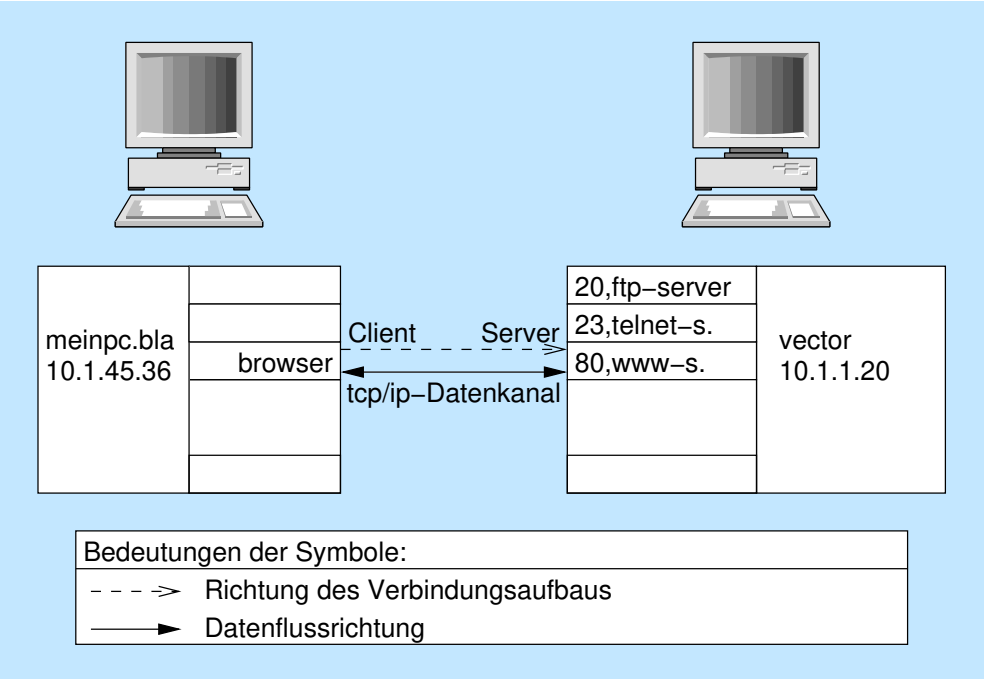


Abbildung 3: Mehrere Server an einem System

```
telnet          23/tcp                  # Telnet
smtp            25/tcp      mail        # Simple Mail Transfer
http            80/tcp      www         # World Wide Web HTTP
pop3           110/tcp                  # Post Office Protocol, Version 3
https          443/tcp                  # http protocol over TLS/SSL
http-alt      8080/tcp                  # HTTP Alternate (Proxy)
##############################################################################
```

### 1.2.2 HTTP-Verbindung per TELNET?

Wie kann man nun herausfinden, ob ein Webserver erreichbar ist?

TELNET ist ein einfaches login-Protokoll in TCP/IP. Vereinfacht dargestellt ist TELNET ein Protokoll, das jedes Zeichen von der Tastatur 1:1 zum fremden Rechner übergibt und seinerseits jedes erhaltene Zeichen auf dem eigenen Rechner darstellt. TELNET eignet sich somit auch zur einfachen Fernwartung fremder Rechner (allerdings naturgemäß im Textmodus).

TELNET weist zwar Sicherheitsmängel auf, ist aber gut geeignet, um andere, kompliziertere Protokolle *zeichenweise nachzubauen* und zu testen. Der Telnet-Client ist auch auf PC-Systemen fast immer verfügbar. Im Gegensatz zu speziellen Werkzeugen (z.B. netcat funktioniert TELNET jedoch nur mit TCP-Protokollen!

Der Aufruf des Telnet-Clients ist z.B.:

Terminal
```
schueler@debian964:~$ telnet localhost 25
```

Mit `localhost` wird der eigene Rechner adressiert (man hätte auch `127.0.0.1` angeben können), 25 ist die Nummer des SMTP-Ports (man hätte auch `smtp` angeben können). Wichtig ist, dass (abweichend von der URL-Notation) zwischen Rechnername und Portnummer kein Doppelpunkt, sondern ein Leerzeichen stehen muss!

### 1.2.3 Mitschnitt einer HTTP-Verbindung

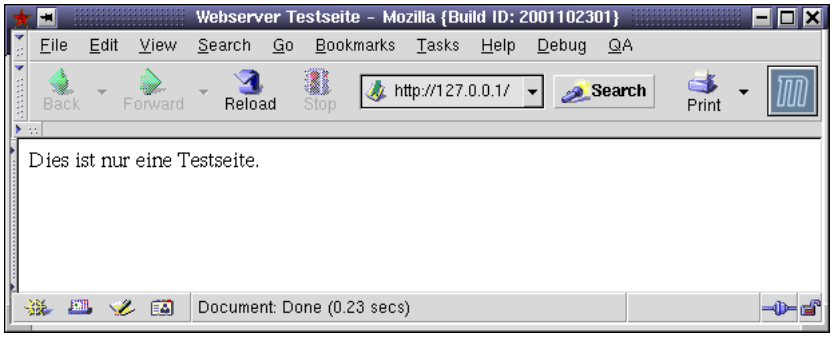


Abbildung 4: HTTP-Verbindung

Hier ist eine recht einfache Beispielverbindung angegeben:

Terminal

```
schueler@debian964:~$ telnet 127.0.0.1 80
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
GET / HTTP/1.0 ↵
↵
HTTP/1.1 200 OK
Date: Sat, 08 Feb 2003 16:38:58 GMT
Server: Apache/1.3.19 (Unix)  (SuSE/Linux)
Connection: close
Content-Type: text/html

<html>
  <head>
    <title>Webserver Testseite</title>
  </head>
  <body>
    Dies ist nur eine Testseite.
  </body>
</html>
Connection closed by foreign host.
```

Die ersten drei Zeilen stammen vom Telnet-Client. Die folgenden zwei Zeilen (`GET` und Leerzeile!) wurden vom Benutzer eingegeben. Der dann folgende Text kann je nach Server anders sein, er wurde offenbar vom Server zum Client geschickt. Die letzte Zeile stammt vom Client und gibt an, dass der Server die Verbindung beendet hat. Die identische Verbindung sieht im Browser aus wie in Abbildung 4 Die ganze Verbindung bestand also aus einer Anfrage (*request*) des Browsers und einer Antwort (*response*) des Servers.

Nach der Antwort folgt (zumindest bei HTTP 0.9 und HTTP 1.0): stets der Abbruch der Verbindung durch den Server. Dieser Ablauf wird graphisch im allgemeinen so dargestellt wie in Abbildung 5.

### 1.2.4 Wie fragt man richtig? Format der Anfrage

Die einfachste Anfrage in HTTP 1.0 sieht also so aus:

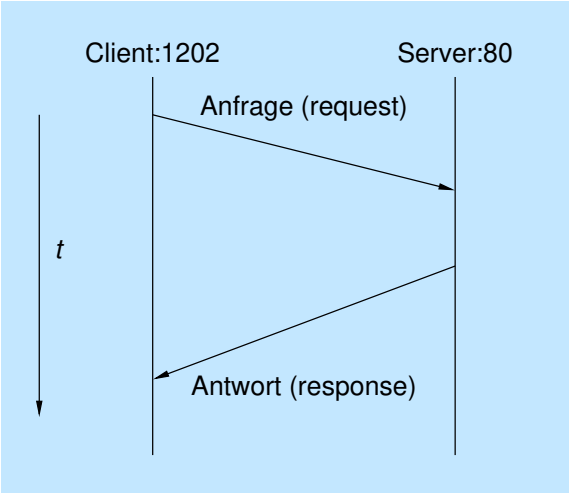


Abbildung 5: HTTP-Verbindung

| Name | Zweck |
|---|---|
| GET | Daten vom Server holen |
| POST | Formulardaten zum Server schicken |
| PUT | Daten auf dem Server speichern |
| DELETE | Daten auf dem Server löschen |

Tabelle 1: HTTP-Anfrage-Methoden

```
GET / HTTP/1.0      # Anfragezeile
                    # Leerzeile
```

Die Leerzeile dient dazu, einen eventuell vorhandenen Anfrage-Rumpf (*body*) vom Anfrage-Kopf (*head*) zu unterscheiden.

Die Anfragezeile (*request line*) hat folgenden Aufbau:

```
Methode URL HTTP/HTTP-Version
```

Tabelle 1 zeigt wichtige Anfrage-Methoden. Die URL (hier gleichbedeutend benutzt mit URI und URN) hat folgenden Aufbau:

```
GET Protokoll://hostname/pfadname
```

Wenn das Protokoll fehlt, wird HTTP angenommen:

```
GET hostname/pfadname
```

Wenn der Hostname fehlt, wird angenommen, dass der eigene Rechner gemeint ist:

```
GET /pfadname
```

Wenn der Pfadname ein Verzeichnisname ist, wird (je nach Einstellung des Webservers) als Dateiname `index.html` angenommen:

```
GET /
```

Zwischen der Anfragezeile und der Leerzeile dürfen noch eine oder mehrere so genannte Headerzeilen stehen, mit dem der Client dem Server weitere Informationen übergeben kann.

### 1.2.5 ...so schallt es heraus — Format der Antwort

In der Antwort des Servers findet man wieder den von der Anfrage her gewohnten Aufbau:

```
Response-Zeile
ggf. Header-Zeilen
                    #Leerzeile
Antwort-Rumpf z.B. in HTML
```

Die Response-Zeile hat dabei den Aufbau:

```
HTTP/HTTP-Version Antwortcode Kommentar
```

Sie lautete in unserem Beispiel:

```
HTTP/1.1 200 OK
```

Die Antwortcodes (hier `200`) dienen dem Browser zur Analyse des Ergebnisses. Sie lassen sich in fünf Gruppen einteilen (siehe Tabelle 2). Der Kommentar (hier `OK`) ist ein Zusatz zur Fehlersuche durch Menschen. Die Antwort kann (wie auch die Anfrage) eine oder mehrere Headerzeilen

| Bereich | Bedeutungen |
|---|---|
| 100 — 199 | Information |
| 200 — 299 | OK |
| 300 — 399 | Umleitung |
| 400 — 499 | Client-Problem |
| 500 — 599 | Server-Problem |

Tabelle 2: HTTP-Antwortcodes

enthalten, die aber hier nicht interessieren.

Das Format des Antwortrumpfes in HTML wird später noch genauer untersucht werden. Es handelt sich offenbar um ein für Menschen lesbares ASCII-Text-Format. Im Text sind Markierungen eingebaut, “Befehle” in spitzen Klammern, die entweder zur Textformatierung oder zur Navigation im Hypertext dienen.