

6 Anwendungsschicht

6.1 Zusammenfassung

Mit der Darstellung der Anwendungsschicht in diesem Kapitel endet die systematische Darstellung der verschiedenen Schichten der Referenzmodelle von unten nach oben. Es werden die wichtigsten Protokolle der Anwendungsschicht beschrieben, die sich im Allgemeinen durch einen relativ einfachen und intuitiven Aufbau auszeichnen.

6.2 Aufgaben und Konzepte

Die **Anwendungsschicht** (*Application Layer*) stellt den Anwendern (Personen), bzw. den Anwendungen (Software) die grundlegenden Eigenschaften der Kommunikation zur Verfügung.

Hierbei ist zu beachten, dass die Anwendung selbst nicht Bestandteil der Anwendungsschicht ist. Zum Beispiel nutzt das Anwendungsprogramm Web-Browser das Protokoll HTTP der Anwendungsschicht und liegt somit oberhalb der Beschreibung des eigentlichen ISO-OSI-Referenzmodells.

Die Zuordnung der Anwendungen zu den Protokollen der Transportschicht erfolgt in der Regel über die Port-Nummer. Eine Übersicht über die wichtigsten Anwendungsprotokolle und deren Verhältnis zur Transportschicht in der IP-Protokollsuite ist in Abbildung 6.1 dargestellt.

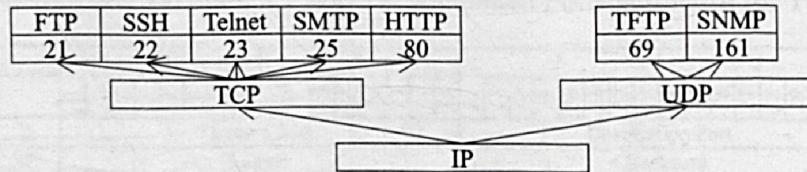


Abbildung 6.1: Die wichtigsten Anwendungsprotokolle in der TCP/IP-Protokollsuite

6.3 Hypertext Transfer Protocol – HTTP

6.3.1 Positionierung

Die heutige Formulierung des *Hypertext Transfer Protocol* wurde 1989 gemeinsam mit der Seitenbeschreibungssprache *Hypertext Markup Language* (HTML) und der Adressenstruktur der *Uniform Resource Locator* (URL) von Tim Berners-Lee und Robert Cailliau am CERN in Genf erarbeitet. Alle drei Bestandteile sind eng miteinander verknüpft und

stellen die Grundlage des *World Wide Web* (WWW) dar (vgl. Abbildung 1.15). Der erste Web-Browser *Mosaic* wurde dann von Studenten der Universität von Illinois programmiert und stellte die ursprüngliche Grundlage für die Produkte der Firma *Netscape Communications Corp.* dar.

Die anfängliche Version HTTP 0.9 war als einfaches Transportprotokoll für Daten ausgelegt. Sie unterstützte lediglich die GET-Methode. Die MIME-Unterstützung für die Übertragung von Binärdaten oder ein Authentifizierungsmechanismus waren nicht vorgesehen. Mit HTTP 1.0 wurde in RFC 1945 ein erweitertes *Request-Response*-Format eingeführt, das die Übermittlung von mehr Informationen in beide Richtungen erlaubt. Dem eigentlichen Request folgt ein Satz von Header-Feldern, die beispielsweise die Übermittlung des Browsetyps erlauben. Seit Version 1.0 kennt HTTP auch die POST-Methode, mit der zum Beispiel Formulareingaben vom Browser zum Web-Server gelangen.

Weitreichendere Verbesserungen brachte 1997 die Einführung von HTTP 1.1, das im RFC 2616 beschrieben ist. Dabei sind insbesondere die persistenten Verbindungen von Interesse. Während bei HTTP 1.0 jeder Request über eine neue Verbindung zum Server übertragen wird, können in der aktuellen Version Verbindungen aufrechterhalten werden, um mehrere Requests zu übermitteln. Über die neue Funktion *Request Pipelining* sendet ein Client nachfolgende Requests an den Server, bevor die Antwort des Servers auf die vorhergehende Anforderung auf der Client-Seite eingetroffen ist. Im Zusammenspiel mit persistenten Verbindungen bringt das einen deutlichen Performance-Gewinn. Ebenfalls für mehr Performance sorgen die in HTTP 1.1 hinzugekommenen Cache-Kontrollfunktionen.

6.3.2 Realisierung

HTTP folgt dem Client-Server-Modell (vgl. Abschnitt 1.3.3). Ein HTTP-Client sendet Anfragen (*Request*) an einen HTTP-Server. Dabei überträgt der Client an den Server-Port 80 (vgl. Abschnitt 5.3.3). Den HTTP-Client bezeichnet man auch als *Web-Browser*, den HTTP-Server auch als *Web-Server*.

HTTP überträgt nicht nur Dateien, sondern Ressourcen (vgl. URL, vgl. Abschnitt 6.3.3). So ist auch die dynamische Antwort auf eine Anfrage oder die Ausgabe eines CGI-Skripts (vgl. Abschnitt 6.3.4.1) möglich. Abbildung 6.2 zeigt den *Request-Response*-Ablauf von HTTP. Die Kommunikation zwischen Client und Server erfolgt durch den Austausch von Nachrichten, die im Wesentlichen aus jeweils zwei Teilen bestehen: aus Header und Daten. Der Header enthält Steuerinformationen, wie z.B. die verwendete Methode und die gewünschte URL. Die HTTP-1.1-Spezifikation sieht insgesamt 46 zum großen Teil optionale Header-Einträge vor. Diese sind in vier Kategorien unterteilt: allgemeine Header-, Response-, Request- und Entity-Header-Einträge. Die allgemeinen Header-Felder sind sowohl in den Anfragen als auch in den Antworten enthalten. Entity-Header beschreiben den Datenteil der Nachricht. Der Datenabschnitt der Nachricht selbst besteht meist aus einem HTML-Dokument oder Formulardaten, die der Client an den Server sendet.

Der Client formuliert dabei seine Anfrage durch die Angabe von Methode, URL und Request-Header-Feldern. Die Methoden sind in Tabelle 6.1 aufgeführt. Der Client muss

einen absoluten URL angeben, damit die Anfrage auch über einen Proxy-Server (vgl. Abschnitt 8.8.3) laufen kann. Nach dem Zugriff auf eine Quelle reicht die Angabe von relativen URLs aus. Ein Request-Header weist folgende Struktur auf:

- > METHOD URL HTTP/version
- > General Header
- > Request Headers
- > Entity Header (optional)
- > Leerzeile
- > Request Entity (falls vorhanden)

Ein Server antwortet auf jede Anfrage, unabhängig davon, ob die gewählte Methode zulässig ist oder nicht. Hierzu dient zunächst eine Status-Meldung, die gemäß der Übersicht aus Tabelle 6.2 codiert ist. Danach folgt die Übertragung der Ressource, falls diese vorhanden ist oder fehlerfrei erzeugt werden kann. Nach der Bearbeitung des Requests beendet der Server die Verbindung. Mit der Ausnahme der *Persistent Connections* aus der Version 1.1 ist HTTP ein zustandsloses Protokoll (*Stateless Protocol*). Der Aufbau einer HTTP-Antwort ist ähnlich dem Aufbau einer Anfrage.

- > HTTP/version Status-Code Reason-Zeile
- > General Header
- > Response Header
- > Entity Header (optional)
- > Leerzeile
- > Resource Entity (falls vorhanden)

Abbildung 6.2 zeigt den Ablauf einer HTTP-Anfrage und Antwort.

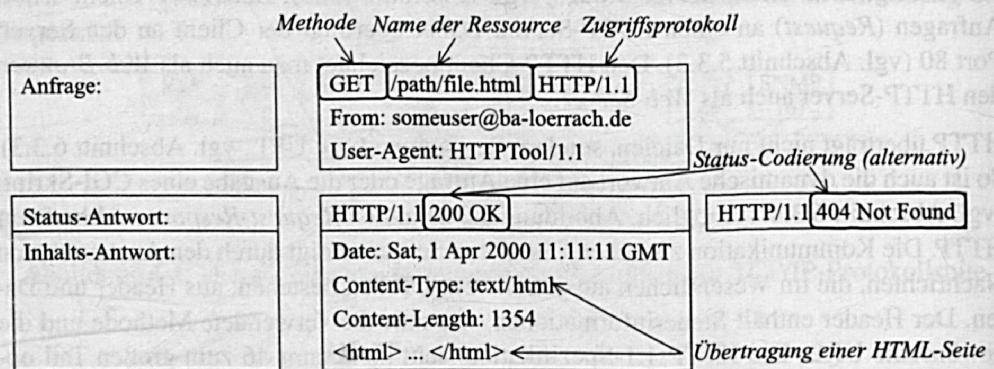


Abbildung 6.2: Ablauf einer HTTP-Anfrage und -Antwort

6.3.3 URL

Der Zugriff auf Ressourcen erfolgt über den sogenannten *Uniform Resource Locator* (URL). Dieser besteht aus den drei in Abbildung 6.3 gezeigten Bestandteilen. Dabei

Methode	Beschreibung
GET	Anfordern einer Ressource vom Server
HEAD	wie GET, es werden aber nur die Header-Informationen der Ressource angefordert
OPTIONS	Informationen über verfügbare Kommunikationsoptionen
POST	Übertragung von Informationen vom Client zum Server (z.B. Formulardaten)
PUT	Modifikation bestehender Quellen bzw. Erzeugung neuer Daten auf dem Server, ähnlich wie POST, aber die URL identifiziert die mit der Anforderung gesendeten Daten selbst und nicht die Quelle
DELETE	Löschen von Daten auf dem Server
TRACE	Verfolgung von Requests, die über mehrere Knotenpunkte (z.B. Proxy-Server) laufen
CONNECT	für Verbindungen, bei denen Proxy-Server dynamisch als Tunnel agieren

Tabelle 6.1: HTTP-Methoden

1xx	indicates an informational message only	100	Continue
2xx	indicates success	200	OK
		201	Created
		202	Accepted
		204	No Content
3xx	redirects the client to another URL	300	Multiple Choices
		301	Moved Permanently
		302	Moved Temporarily
		304	Not Modified
4xx	indicates an error on the client's part	400	Bad Request
		401	Unauthorized
		403	Forbidden
		404	Not Found
5xx	indicates an error on the server's part	500	Internal Server Error
		501	Not Implemented
		502	Bad Gateway
		503	Service Unavailable

Tabelle 6.2: HTTP-Codierung

können folgende Vereinfachungen nützlich sein:

- Bei moderneren Web-Browsern muss die Methode nicht vom Benutzer eingegeben werden. Sie wird dann vom Web-Browser automatisch ergänzt.
- Die Zuordnung von Domänennamen und IP-Adresse erfolgt mit Hilfe des Domain Name System (DNS, vgl. Abschnitt 6.7).
- In den Fällen, in denen man keinen Namen der Ressource angibt, gibt der HTTP-Server eine Seite zurück, die der Web-Master voreingestellt hat. Diese heißt oft "index.html" oder "homepage.htm".

einen absoluten URL angeben, damit die Anfrage auch über einen Proxy-Server (vgl. Abschnitt 8.8.3) laufen kann. Nach dem Zugriff auf eine Quelle reicht die Angabe von relativen URLs aus. Ein Request-Header weist folgende Struktur auf:

- > METHOD URL HTTP/version
- > General Header
- > Request Headers
- > Entity Header (optional)
- > Leerzeile
- > Request Entity (falls vorhanden)

Ein Server antwortet auf jede Anfrage, unabhängig davon, ob die gewählte Methode zulässig ist oder nicht. Hierzu dient zunächst eine Status-Meldung, die gemäß der Übersicht aus Tabelle 6.2 codiert ist. Danach folgt die Übertragung der Ressource, falls diese vorhanden ist oder fehlerfrei erzeugt werden kann. Nach der Bearbeitung des Requests beendet der Server die Verbindung. Mit der Ausnahme der *Persistent Connections* aus der Version 1.1 ist HTTP ein zustandsloses Protokoll (*Stateless Protocol*). Der Aufbau einer HTTP-Antwort ist ähnlich dem Aufbau einer Anfrage.

- > HTTP/version Status-Code Reason-Zeile
- > General Header
- > Response Header
- > Entity Header (optional)
- > Leerzeile
- > Resource Entity (falls vorhanden)

Abbildung 6.2 zeigt den Ablauf einer HTTP-Anfrage und Antwort.

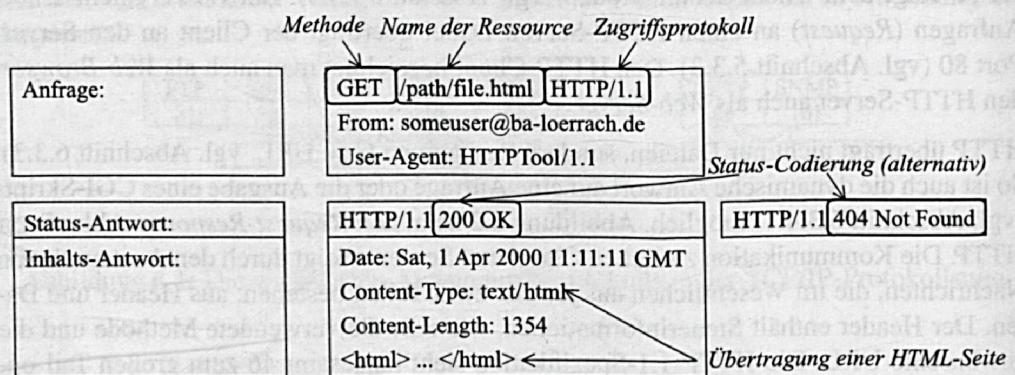


Abbildung 6.2: Ablauf einer HTTP-Anfrage und -Antwort

6.3.3 URL

Der Zugriff auf Ressourcen erfolgt über den sogenannten *Uniform Resource Locator* (URL). Dieser besteht aus den drei in Abbildung 6.3 gezeigten Bestandteilen. Dabei

Methode	Beschreibung
GET	Anfordern einer Ressource vom Server
HEAD	wie GET, es werden aber nur die Header-Informationen der Ressource angefordert
OPTIONS	Informationen über verfügbare Kommunikationsoptionen
POST	Übertragung von Informationen vom Client zum Server (z.B. Formulardaten)
PUT	Modifikation bestehender Quellen bzw. Erzeugung neuer Daten auf dem Server, ähnlich wie POST, aber die URL identifiziert die mit der Anforderung gesendeten Daten selbst und nicht die Quelle
DELETE	Löschen von Daten auf dem Server
TRACE	Verfolgung von Requests, die über mehrere Knotenpunkte (z.B. Proxy-Server) laufen
CONNECT	für Verbindungen, bei denen Proxy-Server dynamisch als Tunnel agieren

Tabelle 6.1: HTTP-Methoden

1xx	indicates an informational message only	100	Continue
		200	OK
2xx	indicates success	201	Created
		202	Accepted
		204	No Content
3xx	redirects the client to another URL	300	Multiple Choices
		301	Moved Permanently
		302	Moved Temporarily
		304	Not Modified
4xx	indicates an error on the client's part	400	Bad Request
		401	Unauthorized
		403	Forbidden
		404	Not Found
5xx	indicates an error on the server's part	500	Internal Server Error
		501	Not Implemented
		502	Bad Gateway
		503	Service Unavailable

Tabelle 6.2: HTTP-Codierung

können folgende Vereinfachungen nützlich sein:

- Bei moderneren Web-Browsern muss die Methode nicht vom Benutzer eingegeben werden. Sie wird dann vom Web-Browser automatisch ergänzt.
- Die Zuordnung von Domänennamen und IP-Adresse erfolgt mit Hilfe des Domain Name System (DNS, vgl. Abschnitt 6.7).
- In den Fällen, in denen man keinen Namen der Ressource angibt, gibt der HTTP-Server eine Seite zurück, die der Web-Master voreingestellt hat. Diese heißt oft "index.html" oder "homepage.htm".

Wie kann auf die Ressource zugegriffen werden?	Wo befindet sich die Ressource?	Wie heißt die Ressource?
Protokoll (Schema)	DNS-Name/IP-Adresse des Rechners, auf dem sich die Ressource befindet	lokaler eindeutiger Name der Seite (normalerweise der Dateiname)
http://	www.ba-loerrach.de 193.196.4.237	/index.html

Abbildung 6.3: Struktur einer URL

6.3.4 Dynamische Ressourcen

6.3.4.1 Server Side Includes – SSI

Web-Server sollten in der Lage sein, dynamische Ressourcen zur Verfügung zu stellen. Hierunter versteht man z.B. die laufzeitabhängige Erzeugung von Dokumenten unter Rückgriff auf Datenbankabfragen. Die Konzepte für dynamische Ressourcen, die auf dem Server laufen und von diesem zur Verfügung gestellt werden, fasst man unter dem Begriff der Server Side Includes (SSI) zusammen:

- Common Gateway Interface (CGI) stellt einen Mechanismus zur Verfügung, um ein Anwendungsprogramm auf dem Server zu starten. Dieses liefert Daten zur Beantwortung der Anfrage, die vom Web-Server in ein HTML-Dokument eingefügt und an den Client übermittelt werden. Ein CGI-Skript ist die Menge der Befehle, die vom Client gewünschte Aktion bewirken.
- Java Servlets sind in Java geschriebene Programme, die auf dem Server ablaufen. Sie werden durch das Tag <SERVLET> in HTML-Dokumente eingebettet.

6.3.4.2 Client Side Includes – CSI

Nicht zu verwechseln hiermit sind Client Side Includes (CSI), die auf dem Client-Rechner ausgeführt werden. Hierzu zählen beispielsweise **Java-Applets**, oder **JavaScript**-Programme, die in HTML-Dokumente integriert sind.

6.4 File Transfer Protocol – FTP

Mit dem File Transfer Protocol (FTP) werden Dateien zwischen einem Client und einem Server über das verbindungsorientierte und zuverlässige TCP auf Transportebene übertragen. FTP benötigt zwei Vollduplexverbindungen. Dabei werden Befehle und Bestätigungen über die Steuerverbindung (Port 21 auf der Server-Seite) übermittelt. Die Datenverbindung zum Datenaustausch greift auf den Server-Port 20 zurück.

Die Steuerverbindung geht vom Client aus. Hierzu nutzt dieser den Befehlssatz, der in Tabelle 6.3 auszugsweise dargestellt ist. Der Server antwortet mit entsprechenden Bestätigungen, die in Tabelle 6.4 beispielhaft aufgeführt sind. Den Ablauf einer FTP-Verbindung zeigt Abbildung 6.4. Die Datenverbindung wird normalerweise, d.h. im so genannten aktiven Modus, vom Server hergestellt. Im so genannten passiven Modus erzeugt hingegen

der Client die Verbindung. Dies kann bei einem Transfer durch Firewall-Systeme nötig sein (vgl. Abschnitt 8.8), wenn diese keine Verbindungen von außerhalb erlauben.

!	beendet die Verbindung temporär und geht auf die Betriebssystemebene zurück
?	Anzeige des Hilfe-Menüs für alle FTP-Kommandos (wie help)
append	fügt ein lokales File in ein Remote File ein
ascii	bewirkt, dass die Files im ASCII-Format transferiert werden
bye	beendet die File-Transfer-Applikation (FTP Program)
ftp	Aufruf des FTP-Kommando-Interpreters
get	überträgt ein File vom Remote-Host auf das lokale System
help	Anzeige des Hilfe-Menüs für alle FTP-Kommandos (wie ?)
put	überträgt ein lokales File auf das Remote-System
quit	beendet das FTP-Programm

Tabelle 6.3: Client-seitige FTP-Befehle

200	Kommando o.k.
220	Rechner bereit für Benutzer
221	Goodbye!
226	Datenverbindung in Abbau, Übertragung abgeschlossen
230	Login o.k.; erwarte Kommando
331	Benutzername o.k.; Passwort gefordert

Tabelle 6.4: Server-seitige FTP-Bestätigungen

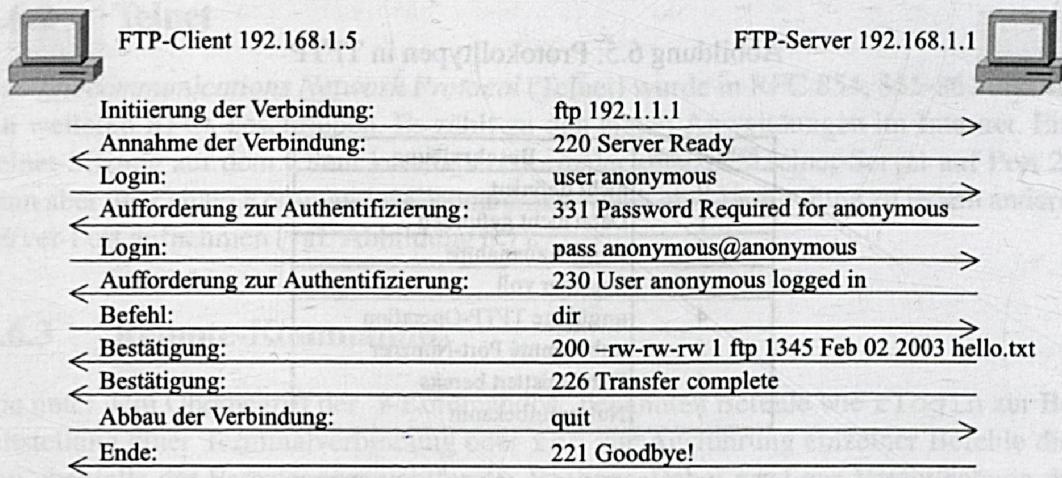


Abbildung 6.4: Aufbau, Betrieb und Abbau einer FTP-Verbindung