10. Foliensatz Computernetze

Prof. Dr. Christian Baun

Frankfurt University of Applied Sciences
(1971–2014: Fachhochschule Frankfurt am Main)
Fachbereich Informatik und Ingenieurwissenschaften
christianbaun@fb2.fra-uas.de

Lernziele dieses Foliensatzes

- Sitzungsschicht
- Darstellungsschicht
- Anwendungsschicht
  - Anwendungsprotokolle
    - Namensauflösung (DNS)
    - Automatische Vergabe von Adressen (DHCP)
    - Zeitsynchronisierung (NTP)
    - Fernsteuerung von Computern (Telnet, SSH)
    - Übertragung von Daten (HTTP)
    - Emails austauschen (SMTP)
    - Emails herunterladen (POP3)
    - Dateien hochladen und herunterladen (FTP)

Sitzungsschicht (Session Layer)

- Verantwortlich für Aufbau, Überwachung und Beenden einer Sitzung
  - Eine Sitzung ist die Grundlage für eine virtuelle Verbindung zwischen zwei Anwendungen auf physisch unabhängigen Rechnern
  - Eine Sitzung besteht aus Anfragen und Antworten zwischen Anwendungen
- Zudem ist der Sitzungsschicht die Dialogkontrolle (welcher Teilnehmer gerade spricht) zugedacht
- Funktionen zur Synchronisierung
  - Kontrollpunkte können in längeren Übertragungen eingebaut werden
  - Kommt es zum Verbindungsabbruch, kann zum letzten Kontrollpunkt zurückgekehrt werden und die Übertragung muss nicht von vorne beginnen

Protokolle der Sitzungsschicht

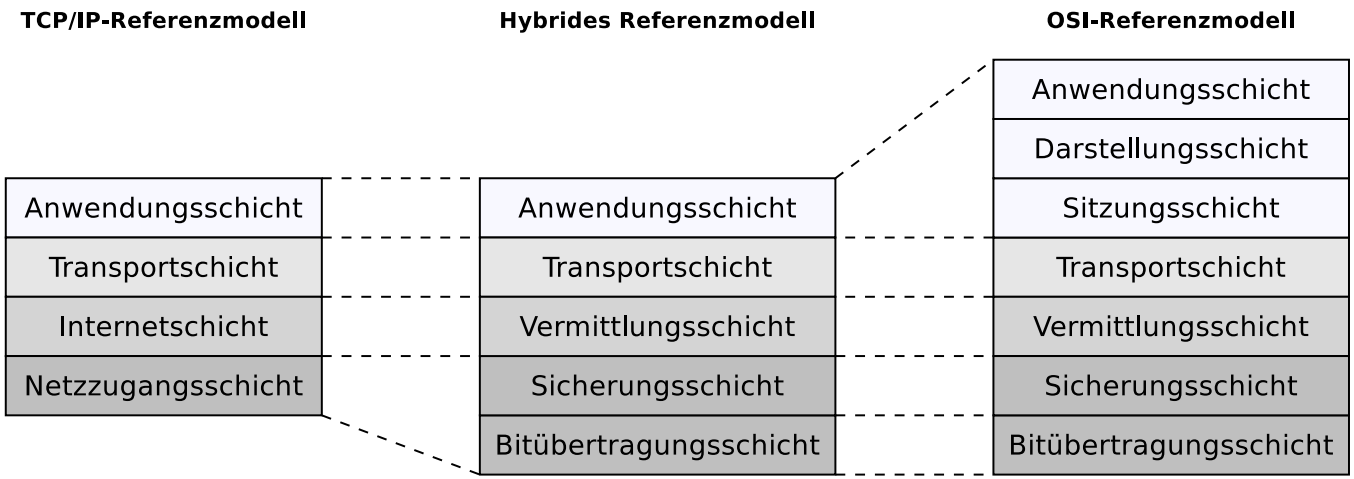
- Protokolle, die den geforderten Fähigkeiten der Sitzungsschicht entsprechen, sind unter anderem Telnet zur Fernsteuerung von Rechnern und FTP zur Übertragung von Dateien
  - Allerdings können diese Protokolle auch der Anwendungsschicht zugeordnet werden
- Die Anwendungsschicht enthält die Protokolle, die die Anwendungsprogramme verwenden
- FTP und Telnet werden direkt von den entsprechenden Anwendungsprogrammen verwendet und nicht von abstrakteren Protokollen in höheren Ebenen
  - Darum is es sinnvoller, die Protokolle der Sitzungsschicht der Anwendungsschicht zuzuordnen

Darstellungsschicht (Presentation Layer)

- Enthält Regeln zur Formatierung (Präsentation) der Nachrichten
  - Der Sender kann den Empfänger informieren, dass eine Nachricht in einem bestimmten Format (zum Beispiel ASCII) vorliegt
    - Ziel: Die eventuell nötige Konvertierung beim Empfänger ermöglichen
  - Datensätze können hier mit Feldern (zum Beispiel Name, Matrikelnummer...) definiert werden
  - Art und Länge der Datentypen können definiert werden
  - Kompression und Verschlüsselung sind der Darstellungsschicht zugedachte Aufgabenbereiche
- Genau wie die Sitzungsschicht wird auch die Darstellungsschicht in der Praxis kaum benutzt
  - Grund: Alle dieser Schicht zugedachten Aufgaben erfüllen heute Anwendungsprotokolle

Anwendungsschicht

- Enthält alle Protokolle, die mit Anwendungsprogrammen (zum Beispiel Browser oder Email-Programm) zusammenarbeiten
- Hier befinden sich die eigentlichen Nachrichten (zum Beispiel HTML-Seiten oder Emails) entsprechend dem jeweiligen Anwendungsprotokoll



- Geräte: keine
- Protokolle: DNS, DHCP, NTP, Telnet, SSH, HTTP, SMTP, FTP...

Domain Name System (DNS)

- Protokoll zur **Namensauflösung** von Domain-Namen zu IP-Adressen

RFC 1034 and 1035

- Analog zur Telefonauskunft
  - Person/Familie/Firma $\Longrightarrow$ Telefonnummer
  - Rechnername/Website $\Longrightarrow$ IP-Adresse

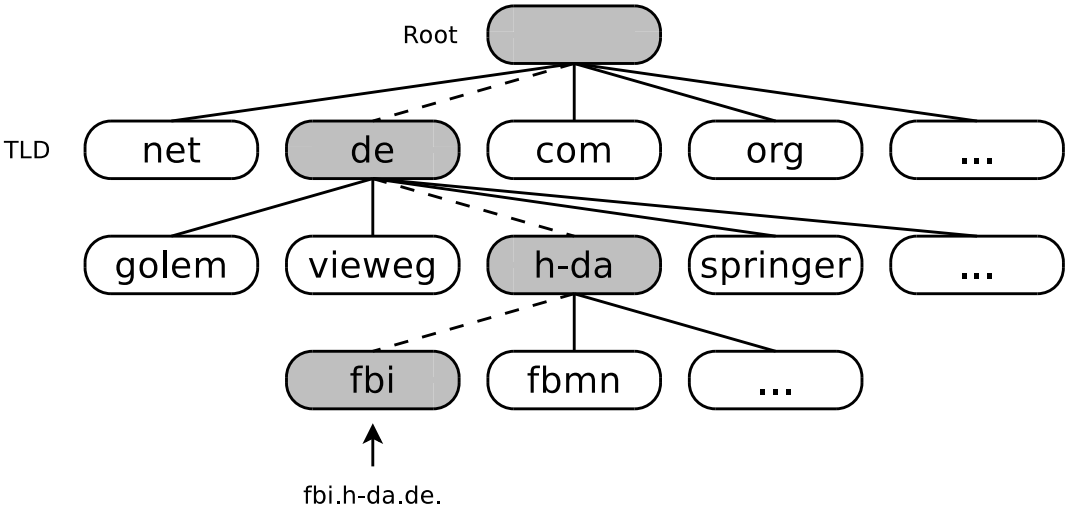
Entwicklung 1983 von Paul Mockapetris

- DNS löste die lokalen Namenstabellen in der Datei `/etc/hosts` ab, die bis dahin für die Verwaltung der Namen/Adressen-Zuordnungen zuständig waren
  - Diese waren der zunehmenden Zahl von Neueinträgen nicht mehr gewachsen
- Basiert auf einem hierarchischen Namensraum
  - Die Information mit den Zuordnungen sind in separate Teile gegliedert und im gesamten Internet auf **Nameservern** verteilt

Domain-Namensraum (1/2)

- Der Domain-Namensraum hat eine **baumförmige Struktur**
  - Die Blätter und Knoten heißen **Labels**
  - Jeder Unterbaum ist eine **Domäne**
- Ein vollständiger Domainname besteht aus der Verkettung aller Labels eines Pfades
- Label sind alphanumerische Zeichenketten
  - Als einziges Sonderzeichen ist der Bindestrich erlaubt
  - Labels sind 1 bis 63 Zeichen lang
  - Labels müssen mit einem Buchstaben beginnen und dürfen nicht mit einem Bindestrich anfangen oder enden
  - Jedes Labels endet mit einem Punkt
- Domainnamen werden mit einem Punkt abgeschlossen
  - Wird meist weggelassen, gehört rein formal aber zu einem vollständigen Domainnamen – **Fully Qualified Domain-Name** (FQDN) dazu
- Ein vollständiger Domainname ist z.B. `www.h-da.de.`

Domain-Namensraum (2/2)



- Domainnamen werden von rechts nach links aufgelöst
  - Je weiter rechts ein Label steht, umso höher steht es im Baum

- Die erste Ebene unterhalb der Wurzel heißt **Top-Level-Domain** (TLD)
- Die DNS-Objekte einer Domäne (zum Beispiel die Rechnernamen) werden als Satz von **Resource Records** (RR) in einer Zonendatei gehalten, die auf einem oder mehreren Nameservern vorhanden ist
- Die Zonendatei heißt häufig einfach **Zone**

Root-Nameserver (1/2)

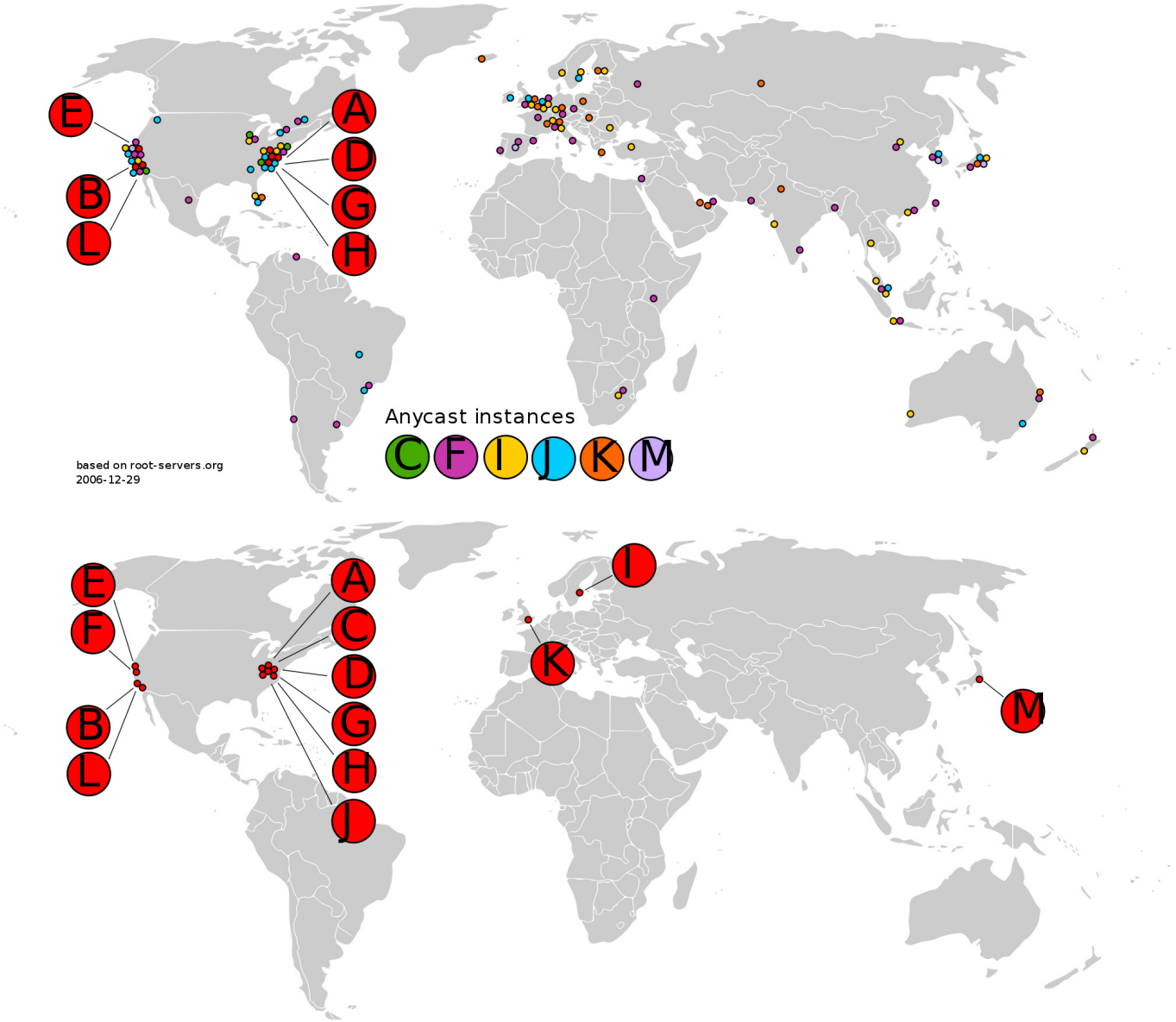
- Die 13 Root-Nameserver (`A` bis `M`) publizieren die **Root-Zone** des DNS
  - Deren Domain-Namen haben die Form `buchstabe.root-servers.net`
  - Die Root-Zone enthält ca. 3000 Einträge und ist die Wurzel des DNS
    - Sie enthält die Namen und IPs der für die TLDs zuständigen Nameserver
- Einige Root-Server bestehen nicht aus einem, sondern mehreren physischen Servern, die zu einem logischen Server verbunden sind
  - Diese Rechner befinden sich an verschiedenen Standorten weltweit und sind via **Anycast** über dieselbe IP-Adresse erreichbar

Name	IPv4-Adresse	IPv6-Adresse	Ort
`A`	198.41.0.4	2001:503:ba3e::2:30	verteilt (Anycast)
`B`	192.228.79.201	—	Kalifornien, USA
`C`	192.33.4.12	—	verteilt (Anycast)
`D`	128.8.10.90	2001:500:2d::d	Maryland, USA
`E`	192.203.230.10	—	Kalifornien, USA
`F`	192.5.5.241	2001:500:2f::f	verteilt (Anycast)
`G`	192.112.36.4	—	verteilt (Anycast)
`H`	128.63.2.53	2001:500:1::803f:235	Maryland, USA
`I`	192.36.148.17	2001:7fe::53	verteilt (Anycast)
`J`	192.58.128.30	2001:503:c27::2:30	verteilt (Anycast)
`K`	193.0.14.129	2001:7fd::1	verteilt (Anycast)
`L`	199.7.83.42	2001:500:3::42	verteilt (Anycast)
`M`	202.12.27.33	2001:dc3::35	verteilt (Anycast)

Root-Nameserver (2/2)

Bildquelle: Wikipedia

- Bevor Anycast eingesetzt wurde, lagen 10 Root-Nameserver in den USA
- Das wurde kritisiert, da diese geographische Zentrierung dem Dezentralisierungsgedanken des Internets entgegenläuft



Aufbau der DNS-Datenbank und Ressourceneinträge

Sie wissen bereits. . .

- DNS ist eine Art verteilte Datenbank mit baumförmiger Struktur
- Beim Internet-DNS liegen die Daten auf einer Vielzahl weltweit verteilter Server, die untereinander über Verweise (*Delegierungen*) verknüpft sind
- In jedem Nameserver existieren ≥ 1 Zonendateien

- Die Zonendateien enthalten Listen von **Resource Records** (RR)
- Jeder RR („*Ressourceneintrag*“) besteht aus 5 Elementen
  `<Name, Wert, Typ, Klasse, TTL>`
- Die Tabelle enthält einige Typen von RRs

Typ	Beschreibung
`NS`	Definiert, welcher Nameserver für die Zone zuständig ist oder verknüpft Zonen zu einem Zonen-Baum (Delegation)
`A`	Enthält die IPv4-Adresse eines Hosts
`AAAA`	Enthält die IPv6-Adresse eines Hosts
`SOA`	Enthält Angaben zur Verwaltung der Zone wie den Namen und die Email-Adresse des Administrators
`CNAME`	Liefert einen Alias-Domain-Namen für einen bestimmten Host
`MX`	Weist einem Namen einen SMTP-Mailserver zu. Alle anderen Dienste nutzen `CNAME`, `A` und `AAAA` Resource Records für die Namensauflösung
`PTR`	Weist einer IP-Adresse einen oder mehrere Hostname(s) zu. Gegenstück zur üblichen Zuordnung einer oder mehrerer IPs zu einem Hostnamen per `A` oder `AAAA` Resource Record

Beispiel einer Namensauflösung (1/5)

- Im folgenden Beispiel wird der Namen `www.fh-frankfurt.de.` mit dem Kommandozeilenwerkzeug `dig` aufgelöst

```
dig +trace +additional -t A www.fh-frankfurt.de.
```

- `-t A` $\Longrightarrow$ `A` Resource Record (die IPv4-Adresse) anfragen
- `+trace` $\Longrightarrow$ Die einzelnen Antworten auf dem Pfad durch die Nameserver-Hierarchie ausgeben
- `+additional` $\Longrightarrow$ Nameserver verwalten für Delegierungen nicht nur `NS` Resource Records, sondern teilweise auch deren IP-Adressen in Form von `A` oder `AAAA` RRs. Diese Option sorgt dafür, dass sie mit ausgeben werden

- Auf dem Weg zur IP müssen nacheinander 4 Nameserver befragt werden

Beispiel einer Namensauflösung (2/5)

```
$ dig +trace +additional -t A www.fh-frankfurt.de.

; <<>> DiG 9.7.1-P2 <<>> +trace +additional -t A www.fh-frankfurt.de.
;; global options: +cmd
.                       3600000 IN      NS      a.root-servers.net.
.                       3600000 IN      NS      l.root-servers.net.
.                       3600000 IN      NS      j.root-servers.net.
.                       3600000 IN      NS      b.root-servers.net.
.                       3600000 IN      NS      c.root-servers.net.
.                       3600000 IN      NS      f.root-servers.net.
.                       3600000 IN      NS      d.root-servers.net.
.                       3600000 IN      NS      e.root-servers.net.
.                       3600000 IN      NS      k.root-servers.net.
.                       3600000 IN      NS      h.root-servers.net.
.                       3600000 IN      NS      i.root-servers.net.
.                       3600000 IN      NS      g.root-servers.net.
.                       3600000 IN      NS      m.root-servers.net.
;; Received 241 bytes from 10.0.0.1#53(10.0.0.1) in 46 ms
```

- In der letzten Zeile ist 10.0.0.1 die IP des Nameservers des abfragenden Rechners
  - Dieser Nameserver kennt die IP-Adressen der Root-Nameserver
  - Die IP-Adressen der Root-Nameserver ändern sich sehr selten und müssen allen Nameservern bekannt sein, sofern sie das Internet betreffende Anfragen beantworten

Beispiel einer Namensauflösung (3/5)

```
de.                     172800  IN      NS      a.nic.de.
de.                     172800  IN      NS      f.nic.de.
de.                     172800  IN      NS      l.de.net.
de.                     172800  IN      NS      n.de.net.
de.                     172800  IN      NS      s.de.net.
de.                     172800  IN      NS      z.nic.de.
a.nic.de.               172800  IN      A       194.0.0.53
f.nic.de.               172800  IN      A       81.91.164.5
l.de.net.               172800  IN      A       77.67.63.105
n.de.net.               172800  IN      A       194.146.107.6
s.de.net.               172800  IN      A       195.243.137.26
z.nic.de.               172800  IN      A       194.246.96.1
a.nic.de.               172800  IN      AAAA    2001:678:2::53
f.nic.de.               172800  IN      AAAA    2a02:568:0:2::53
l.de.net.               172800  IN      AAAA    2001:668:1f:11::105
n.de.net.               172800  IN      AAAA    2001:67c:1011:1::53
;; Received 351 bytes from 199.7.83.42#53(l.root-servers.net) in 79 ms
```

- Aus den 13 Root-Nameservern wurde zufällig `l.root-servers.net` ausgewählt, um ihm die Frage nach `www.fh-frankfurt.de.` zu stellen
- Die Antwort enthält 6 Nameserver zur Auswahl, die für die Zone `de.` verantwortlich sind
  - Bei 4 Servern ist die Abfrage auch mittels IPv6 (`AAAA`) möglich

Beispiel einer Namensauflösung (4/5)

```
fh-frankfurt.de.          86400   IN      NS      deneb.dfn.de.
fh-frankfurt.de.          86400   IN      NS      medusa.fh-frankfurt.de.
fh-frankfurt.de.          86400   IN      NS      chaplin.rz.uni-frankfurt.de.
deneb.dfn.de.             86400   IN      A       192.76.176.9
medusa.fh-frankfurt.de.   86400   IN      A       192.109.234.209
chaplin.rz.uni-frankfurt.de. 86400 IN     A       141.2.22.74
;; Received 169 bytes from 77.67.63.105#53(l.de.net) in 54 ms
```

- Aus den 6 genannten Nameservern wurde zufällig `l.de.net` ausgewählt, um ihm die Frage nach `www.fh-frankfurt.de.` zu stellen
- Die Antwort enthält 3 möglichen Delegierungen zur Auswahl, die für die Zone `fh-frankfurt.` verantwortlich sind

Beispiel einer Namensauflösung (5/5)

```
www.fh-frankfurt.de.     86400   IN      CNAME   squid01.dv.fh-frankfurt.de.
squid01.dv.fh-frankfurt.de. 86400 IN     A       192.109.234.216
fh-frankfurt.de.         86400   IN      NS      medusa.fh-frankfurt.de.
fh-frankfurt.de.         86400   IN      NS      deneb.dfn.de.
medusa.fh-frankfurt.de. 86400    IN      A       192.109.234.209
;; Received 139 bytes from 192.109.234.209#53(medusa.fh-frankfurt.de) in 57 ms
```

- Aus den 3 genannten Nameservern wurde zufällig `medusa.fh-frankfurt.de` ausgewählt, um ihm die Frage nach `www.fh-frankfurt.de.` zu stellen
- Ergebnis: 192.109.234.216

Protokoll von DNS

- DNS-Anfragen werden meist per UDP Port 53 zum Namensserver gesendet
- Die maximal zulässige Länge einer DNS-Antwort via UDP beträgt 512 Bytes
- Längere DNS-Antworten sendet der Nameserver via TCP

Dynamic Host Configuration Protocol (DHCP)

- Ermöglicht die Zuweisung der Netzwerkkonfiguration (IP-Adresse, Netzmaske, Default-Gateway, Nameserver, usw.) an Netzwerkgeräte mit Hilfe eines **DHCP-Clients** durch einen **DHCP-Server**
  - Speziell bei mobilen Geräten ist es nicht sinnvoll, feste IPs zu vergeben
    - Bei Änderungen an der Topologie des Netzes müsste man ansonsten auf allen Clients die Netzwerkeinstellungen anpassen
    - Bei DHCP wird nur die Konfiguration des DHCP-Servers angepasst
- Verwendet UDP via Ports 67 (Server oder Relay-Agent) und 68 (Client)

RFC 2131

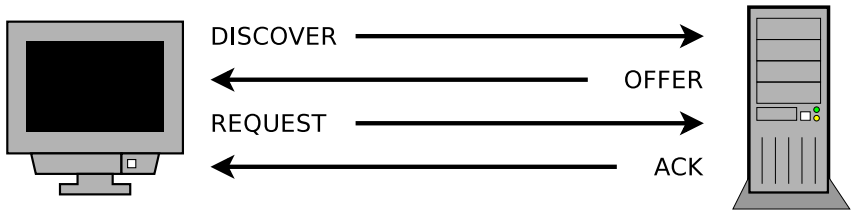
- Ein DHCP-Server verfügt über einen **Pool an IPs** und verteilt diese an Clients
- Damit ein DHCP-Client einen DHCP-Server nutzen kann, muss sich dieser **im selben logischen Netz** befinden
  - Grund: DHCP verwendet **Broadcasts** und Router leiten diese nicht weiter

Liegt der DHCP-Server in einem anderen logischen Netz, muss ein **DHCP-Relay** die Anfragen an den DHCP Server weiterleiten

Arbeitsweise von DHCP (1/2)

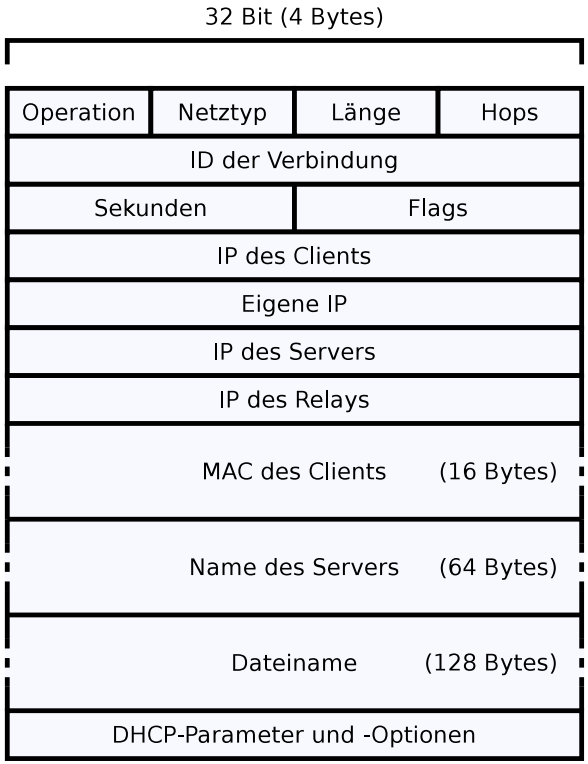
1. Ein Client ohne IP-Adresse sendet als **Broadcast** eine Anfrage (**DHCP-Discover**) an die erreichbaren DHCP-Server
   - Es kann mehrere DHCP-Server in einem Subnetz geben
   - Die Absender-IP-Adresse des Broadcast ist 0.0.0.0
   - Die Zieladresse ist 255.255.255.255
2. Jeder erreichbare DHCP-Server mit freien IP-Adressen in seinem Pool antwortet auf die Anfrage mit einem Adressangebot (**DHCP-Offer**)
   - Auch das Adressangebot wird als **Broadcast** mit der Zieladresse 255.255.255.255 gesendet
3. Der DHCP-Client nimmt ein Adressangebot an, indem er eine Anfrage (**DHCP-Request**) via **Broadcast** ins Netzwerk schickt
   - Die Nachricht enthält die ID des gewünschten DHCP-Servers
   - Die (eventuell vorhandenen) weiteren DHCP-Server erkennen in der Nachricht die Absage für ihre Adressangebote
4. Der Server antwortet mit **DHCP-Ack** und markiert die IP-Adresse in seinem Adresspool als vergeben
   - Er kann die Anfrage auch mit **DHCP-Nak** ablehnen

Arbeitsweise von DHCP (2/2)



- Hat ein DHCP-Server eine Adresse vergeben und dies mit **DHCP-Ack** bestätigt, trägt er in seiner Datenbank bei der Adresse ein *Lease* ein
  - Sind alle Adressen vergeben (verliehen), können keine weiteren Clients mit IP-Adressen versorgt werden
- Jede Adresse besitzt ein Verfallsdatum (*Lease Time*)
  - Dieses wird mit der Bestätigung (**DHCP-Ack**) an den Client übermittelt
  - Aktive Clients verlängern den Lease regelmäßig nach der Hälfte der Lease-Zeit mit einem erneuten **DHCP-Request** direkt via **Unicast** an den Server und nicht per Broadcast
  - Der Server antwortet mit einer erneuten Bestätigung (**DHCP-Ack**) mit den identischen Daten wie vorher und einem neuen Verfallsdatum
  - Ist das Verfallsdatum abgelaufen, kann der Server die Adresse bei Anfragen neu vergeben

Aufbau von DHCP-Nachrichten



- **Operation** legt fest, um was für eine DHCP-Nachricht es sich handelt
  - 1 = Anforderung (*Request*) eines Clients
  - 2 = Antwort (*Reply*) eines Servers
- **Netztyp** gibt die Vernetzungstechnologie an
  - 1 = Ethernet, 6 = WLAN
- **Länge** definiert die Länge der physischen Netzadresse in Bytes
- **Hops** ist optional und gibt die Anzahl der DHCP-Relays auf dem Pfad an
- **Flags** gibt an, ob der Client noch eine gültige IP-Adresse hat
- **Dateiname** ist optional und enthält den Namen einer Datei, die sich der Client via Trivial File Transfer Protocol (TFTP) holen soll
  - Damit kann ein Endgerät über das Netzwerk booten

Network Time Protocol (NTP)

- Standard zur Synchronisierung von Uhren zwischen Computersystemen

RFC 5905 beschreibt das Protokoll und die Algorithmen im Detail

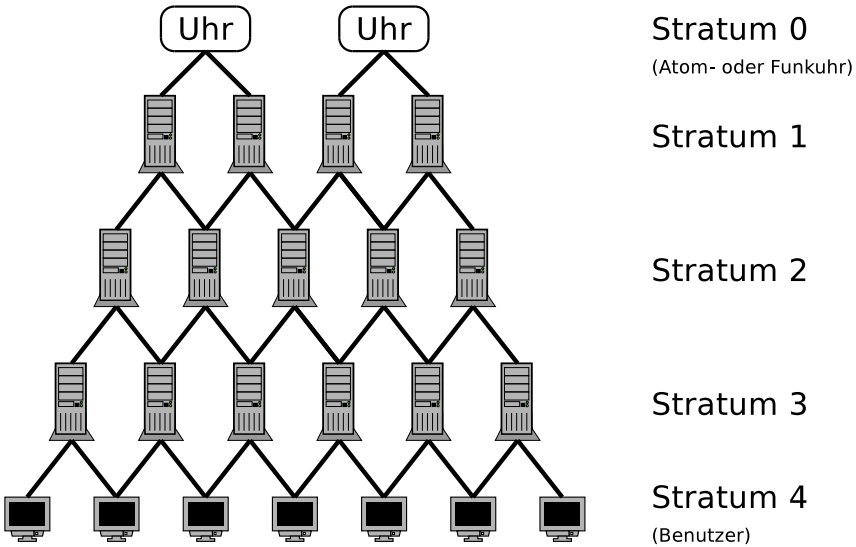
- NTP steht für das Protokoll und für die Referenzimplementierung
  - Verwendet UDP via Port 123

Entwickelt im 1985 von David L. Mills an der Universität von Delaware

- Die lokale Uhr wird vom lokalen Hintergrundprozess (Dämon) der NTP-Software mit einem externen Zeitsignal (z.B. Atom-Uhr, lokaler Funkempfänger oder entfernter NTP-Server via NTP) synchronisiert
- Die Zeitstempel im NTP sind 64 Bit lang
  - 32 Bit enthalten die *UNIX-Zeit* (Sekunden seit dem 1. Januar 1970 00:00:00 Uhr)
  - 32 Bit enthalten den Sekundenbruchteil
  - Ein Zeitraum von 2^{32} Sekunden (ca. 136 Jahre) mit einer Auflösung von 2^{-32} Sekunden (ca. 0,23 Nanosekunden) ist so darstellbar

Hierarchische Struktur eines Verbundes von NTP-Servern

- NTP nutzt ein hierarchisches System sogenannter *Strata*
  - Stratum 0 ist eine Atomuhr oder Funkuhr auf Basis des Zeitsignalsenders DCF77 oder des globalen Navigationssatellitensystems (GPS)
  - Stratum 1 sind die direkt mit Stratum 0 gekoppelten NTP-Server (*Zeitserver*)
  - Darunter folgen weitere Ebenen und die Endgeräte
  - Die Stratum-Ebene gibt den Abstand von Stratum 0 an



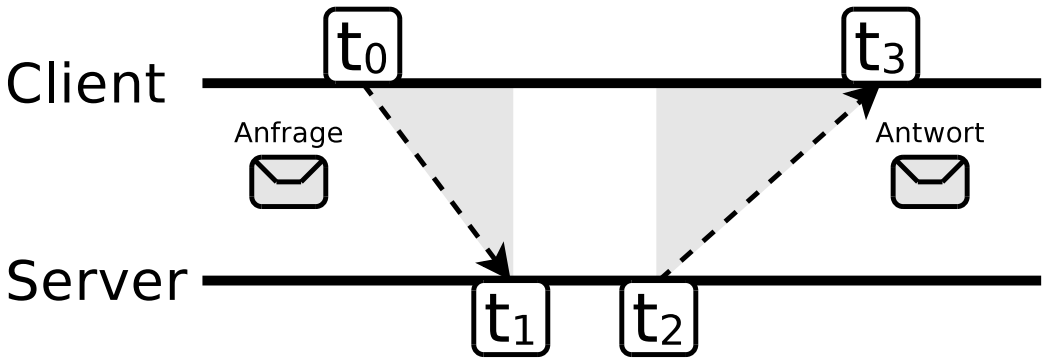
- Die NTP-Software auf Stratum 1, 2, usw. ist zugleich Client des darüber liegenden Stratums als auch Server für das darunter liegende Stratum, wenn es denn existiert
- NTP verwendet die UTC-Zeitskala
- $>$ 100.000 NTP-Knoten existierten weltweit

Eine NTP-Zeitquelle (Stratum 0)

U.S. Naval Observatory – Schriever Air Force Base in Colorado
Bildquelle: `http://www.af.mil/shared/media/photodb/photos/060104-F-3966R-005.jpg`

Zeit-Synchronisations-Algorithmus von NTP

- Um die lokale Uhr mit einem NTP-Server zu synchronisieren, muss ein NTP-Client, die Umlaufzeitverzögerung und die Abweichung berechnen
  - Zeitpunkt t_0: Client sendet Anfrage
  - Zeitpunkt t_1: Server empfängt Anfrage
  - Zeitpunkt t_2: Server sendet Antwort
  - Zeitpunkt t_3: Client empfängt Antwort
  - $t_3 - t_0 \Longrightarrow$ Zeitraum zwischen Senden und Empfangen des Clients
  - $t_2 - t_1 \Longrightarrow$ Zeitraum zwischen Empfangen und Senden des Servers



- Umlaufzeitverzögerung (Round Trip Delay Time) $= (t_3 - t_0) - (t_2 - t_1)$
- Abweichung (Offset) $= \frac{(t_1 - t_0) + (t_2 - t_3)}{2}$

Output of the NTP Daemon

- Meist fragt ein Client ≥ 3 NTP-Server in verschiedenen Netzen ab
  - Ausreißer werden verworfen
  - Eine geschätzte Abweichung (Offset) wird aus den besten Kandidaten berechnet

```
$ ntpq -p
     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
+foxtrot.zq1.de  235.106.237.243  3 u  247 1024  277    49.765   -2.701  46.993
*ns2.customer-re 40.33.41.76      2 u  331 1024  377    50.853    0.390 234.340
+nono.com        78.46.60.42      3 u  746 1024  377    50.469    0.307  28.140
+thw23.de        52.239.121.49    3 u  969 1024  377    51.589    0.308  58.305
```

- Spalte 1: DNS-Name des NTP-Servers
- Spalte 2: IP des NTP-Servers
- Spalte 3: Stratum des NTP-Servers
- Spalte 4: Typ des NTP-Servers (`u` = Unicast)
- Spalte 5: Vergangene Sekunden seit der letzten Anfrage
- Spalte 6: Anfrageintervall in Sekunden
- Spalte 7: Wie häufig wurde der NTP-Server erfolgreich erreicht (377 = die letzten 8 mal)
- Spalte 8: `delay` = Round Trip Time
- Spalte 9: `offset` der lokalen Uhr gegenüber dem NTP-Server
- Spalte 10: `jitter` = Genauigkeitsschwankungen im Übertragungstakt

Telnet (Telecommunication Network)

- Protokoll (RFC 854) zur Fernsteuerung von Rechnern
  - Ermöglicht zeichenorientierten Datenaustausch über **TCP** via Port 23
  - Eignet sich nur für Anwendungen ohne grafische Benutzeroberfläche
- Software, die das Protokoll implementiert, heißt auch einfach Telnet
  - Besteht aus Telnet-Client und Telnet-Server
- Nachteil: **Keine Verschlüsselung!**
  - Auch die Passwörter werden im Klartext versendet
    $\Longrightarrow$ zu unsicher für entferntes Arbeiten
  - Nachfolger: Secure Shell (SSH)
- Wird häufig zur Fehlersuche bei anderen Diensten, zum Beispiel Web-Servern, FTP-Servern oder SMTP-Servern, und zur Administration von Datenbanken sowie in LANs eingesetzt
- Telnet-Clients können sich **mit beliebigen Portnummern verbinden**
  - Das ermöglicht dem Administrator, über einen Telnet-Client, Kommandos an Web-Server, FTP-Server oder SMTP-Server zu senden und unverfälscht deren Reaktion zu beobachten

Telnet und das virtuelle Netzwerkterminal

- Telnet basiert auf dem Standard NVT
  - NVT (Network Virtual Terminal) = virtuelles Netzwerkterminal
    - Herstellerunabhängige Schnittstelle
    - Konvertierungskonzept für unterschiedliche Datenformate
    - Wird von allen Telnet-Implementierungen auf allen Hardwareplattformen unterstützt
    - Ein NVT besteht aus einem Eingabegerät und einem Ausgabegerät, die jeweils nur bestimmte Zeichen erzeugen bzw. anzeigen können
    - Telnet-Clients konvertieren die Tasteneingaben und Kontrollanweisungen in das NVT-Format und übertragen diese Daten an den Telnet-Server, der sie wiederrum dekodiert und weiterreicht
- NVT arbeitet mit Informationseinheiten von je 8 Bits (1 Byte)
- NVT verwendet die 7-Bit-Zeichenkodierung US-ASCII
- Das höchstwertige Bit jedes Zeichens wird mit Null aufgefüllt, um auf 8 Bits zu kommen

Kontrollanweisungen in Telnet

- Die Tabelle enthält die Kontrollanweisungen von NVT
  - Die ersten 3 Kontrollzeichen versteht jeder Telnet-Client und -Server
  - Die übrigen 5 Kontrollzeichen sind optional

Name	Code	Beschreibung
NULL	`NUL`	No operation
Line Feed	`LF`	Zeilenvorschub (nächste Zeile, gleiche Spalte)
Carriage Return	`CR`	Wagenrücklauf (gleiche Zeile, erste Spalte)
BELL	`BEL`	Hörbares oder sichtbares Signal
Back Space	`BS`	Cursor eine Position zurück bewegen
Horizontal Tab	`HT`	Horizontaler Tabulatorstopp
Vertical Tab	`VT`	Vertikaler Tabulatorstopp
Form Feed	`FF`	Cursor in die erste Spalte der ersten Zeile bewegen und Terminal löschen

Secure Shell (SSH)

- Ermöglicht eine verschlüsselte und damit sichere Verbindung zwischen 2 Rechnern über ein unsicheres Netzwerk
  - Sichere Alternative zu Telnet
- Verwendet TCP und standardmäßig Port 22

- SSH-1 wurde 1995 von Tatu Ylönen entwickelt und als Freeware veröffentlicht
- Quelloffene Alternative: OpenSSH (`http://openssh.com`)
- SSH-2 wurde 1996 veröffentlicht und hat u.a. eine verbesserte Integritätsprüfung

- X11 kann über SSH transportiert werden
- Beliebige TCP/IP-Verbindungen können über SSH getunnelt werden (Port-Weiterleitung)
  - Häufige Anwendung: X11-Anwendungen via SSH tunneln
  - Sichere Alternative zu Telnet
- SSH-2 verwendet den Verschlüsselungsalgorithmus AES mit 128 Bit Schlüssellänge
  - Zudem werden 3DES, Blowfish, Twofish, CAST, IDEA, Arcfour, SEED und AES mit anderen Schlüssellängen unterstützt

Hypertext-Übertragungsprotokoll (HTTP)

- Das Hypertext Transfer Protocol (HTTP) ist ein zustandsloses Protokoll zur Übertragung von Daten
  - Zustandslos heißt, dass jede HTTP-Nachricht alle nötigen Informationen enthält, um die Nachricht zu verstehen
  - Der Server hält keine Zustands- bzw. Sitzungsinformation über den Client vor, und jede Anfrage ist eine von anderen Anfragen unabhängige Transaktion

HTTP

Ab 1989 von Roy Fielding, Tim Berners-Lee und anderen am CERN entwickelt

- Ist gemeinsam mit den Konzepten URL und HTML die Grundlage des World Wide Web (WWW)
- Haupteinsatzzweck: Webseiten aus dem World Wide Web (WWW) in einen Browser laden
- Zur Kommunikation ist HTTP auf ein zuverlässiges Transportprotokoll angewiesen
  - In den allermeisten Fällen wird TCP verwendet
- Jede HTTP-Nachricht besteht aus:
  - Nachrichtenkopf (*HTTP-Header*): Enthält u.a. Informationen zu Kodierung, gewünschter Sprache, Browser und Inhaltstyp
  - Nachrichtenkörper (*Body*): Enthält die Nutzdaten, wie den HTML-Quelltext einer Webseite

HTTP-Anfragen (1/2)

- Wird via HTTP auf eine URL (z.B. `http://www.informatik.hs-mannheim.de/~baun/index.html` zugegriffen, wird an den Rechner mit dem Hostnamen `www.informatik.hs-mannheim.de` eine Anfrage für die Ressource `/~baun/index.html` gesendet
- Zuerst wird der Hostname via DNS in eine IP-Adresse umgewandelt
- Über TCP wird zu Port 80, auf dem der Web-Server üblicherweise arbeitet, folgende HTTP-`GET`-Anforderung gesendet

```
GET /~baun/index.html HTTP/1.1
Host: www.informatik.hs-mannheim.de
User-Agent: Mozilla/5.0 (X11; U; Linux i686; de; rv:1.9.2.18) Gecko/20110628 Ubuntu/10.10 (
     maverick) Firefox/3.6.18
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: de-de,de;q=0.8,en-us;q=0.5,en;q=0.3
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 115
Connection: keep-alive
...
```

HTTP-Anfragen (2/2)

- So ein großer Nachrichtenkopf ist eigentlich nicht nötig
- Die hier angegebene HTTP-`GET`-Anforderung genügt völlig

```
GET /~baun/index.html HTTP/1.1
Host: www.informatik.hs-mannheim.de
```

- Der Nachrichtenkopf einer HTTP-Nachricht wird mit einem Line Feed (`LF`) und einem Carriage Return (`CR`) vom Nachrichtenkörper abgegrenzt
  - Im Beispiel hat die HTTP-Anforderung aber keinen Nachrichtenkörper

HTTP-Antworten (1/2)

- Die HTTP-Antwort des Web-Servers besteht aus einem Nachrichtenkopf und dem Nachrichtenkörper mit der eigentlichen Nachricht
  - In diesem Fall enthält der Nachrichtenkörper den Inhalt der angeforderten Datei `index.html`

```
HTTP/1.1 200 OK
Date: Sun, 04 Sep 2011 15:19:13 GMT
Server: Apache/2.2.17 (Fedora)
Last-Modified: Mon, 22 Aug 2011 12:37:04 GMT
ETag: "101ec1-2157-4ab17561a3c00"
Accept-Ranges: bytes
Content-Length: 8535
Keep-Alive: timeout=13, max=499
Connection: Keep-Alive
Content-Type: text/html

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
        "http://www.w3.org/TR/html4/loose.dtd">
<html>
...
</html>
```

HTTP-Antworten (2/2)

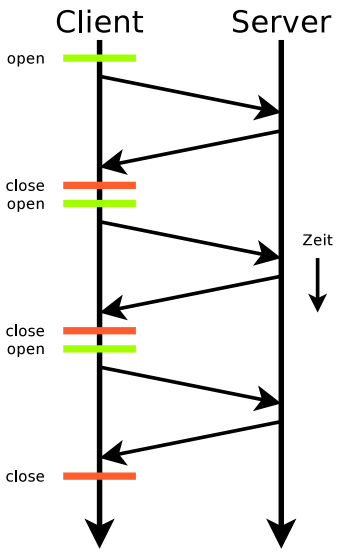
- Jede HTTP-Antwort enthält einen **Statuscode**, der aus 3 Ziffern besteht, und eine Textkette, die den Grund für die Antwort beschreibt

Statuscode	Bedeutung	Beschreibung
1xx	**Informationen**	Anfrage erhalten, Prozess wird fortgeführt
2xx	**Erfolgreiche Operation**	Aktion erfolgreich empfangen
3xx	**Umleitung**	Weitere Aktion des Clients erforderlich
4xx	**Client-Fehler**	Anfrage des Clients fehlerhaft
5xx	**Server-Fehler**	Fehler, dessen Ursache beim Server liegt

- Die Tabelle enthält einige bekannte Statuscodes von HTTP

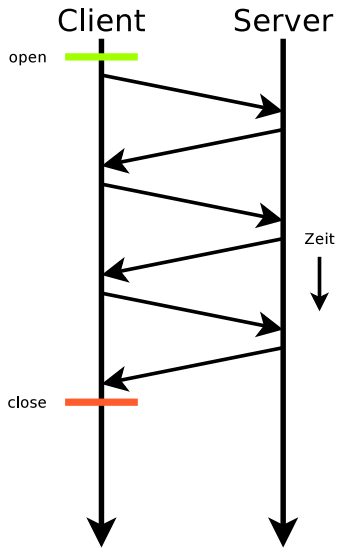
Statuscode	Bedeutung	Beschreibung
200	`OK`	Anfrage erfolgreich bearbeitet. Ergebnis wird in der Antwort übertragen
202	`Accepted`	Anfrage akzeptiert, wird aber zu einem späteren Zeitpunkt ausgeführt
204	`No Content`	Anfrage erfolgreich durchgeführt. Antwort enthält bewusst keine Daten
301	`Moved Permanently`	Ressource verschoben. Die alte Adresse ist nicht länger gültig
307	`Temporary Redirect`	Ressource verschoben. Die alte Adresse bleibt gültig
400	`Bad Request`	Anfrage war fehlerhaft aufgebaut
401	`Unauthorized`	Anfrage kann nicht ohne gültige Authentifizierung durchgeführt werden
403	`Forbidden`	Anfrage mangels Berechtigung des Clients nicht durchgeführt
404	`Not Found`	Ressource vom Server nicht gefunden
500	`Internal Server Error`	Unerwarteter Serverfehler

HTTP/1.0 und HTTP/1.1



- 2 Protokollversionen existieren: `HTTP/1.0` und `HTTP/1.1`
- HTTP/1.0 (RFC 1945): Vor jeder Anfrage wird eine neue TCP-Verbindung aufgebaut und nach der Übertragung der Antwort standardmäßig vom Server wieder geschlossen
  - Enthält ein HTML-Dokument Referenzen auf zum Beispiel 10 Bilder, sind also 11 TCP-Verbindungen zur Übertragung an den Client nötig

- HTTP/1.1 (RFC 2616): Es wird standardmäßig kein Verbindungsabbau durchgeführt
  - So kann die Verbindung immer wieder verwendet werden
  - Für den Transfer eines HTML-Dokuments mit 10 Bildern ist somit nur eine einzige TCP-Verbindung nötig
    - Dadurch wird das Dokument schneller geladen
  - Zudem können abgebrochene Übertragungen bei `HTTP/1.1` fortgesetzt werden



HTTP-Methoden

- Das HTTP-Protokoll enthält einige Methoden für Anfragen

HTTP	Beschreibung
`PUT`	Neue Ressource auf den Web-Server hochladen
`GET`	Ressource vom Web-Server anfordern
`POST`	Daten zum Web-Server hochladen, um Ressourcen zu erzeugen
`DELETE`	Eine Ressource auf dem Web-Server löschen
`HEAD`	Header einer Ressource vom Web-Server anfordern, aber nicht den Body
`TRACE`	Liefert die Anfrage so zurück, wie der Web-Server sie empfangen hat. Hilfreich für die Fehlersuche
`OPTIONS`	Liste der vom Web-Server unterstützten HTTP-Methoden anfordern
`CONNECT`	SSL-Tunnel mit einem Proxy herstellen

HTTP ist ein zustandsloses Protokoll. Über Cookies in den Header-Informationen sind dennoch Anwendungen realisierbar, die Status- bzw. Sitzungseigenschaften erfordern weil sie Benutzerinformationen oder Warenkörbe den Clients zuordnen.

Eine Möglichkeit, Web-Server zu testen, ist telnet (1/2)

```
$ telnet www.informatik.hs-mannheim.de 80
Trying 141.19.145.2...
Connected to anja.ki.fh-mannheim.de.
Escape character is '^]'.
GET /~baun/index.html HTTP/1.0

HTTP/1.1 200 OK
Date: Sun, 04 Sep 2011 21:43:53 GMT
Server: Apache/2.2.17 (Fedora)
Last-Modified: Mon, 22 Aug 2011 12:37:04 GMT
ETag: "101ec1-2157-4ab17561a3c00"
Accept-Ranges: bytes
Content-Length: 8535
Connection: close
Content-Type: text/html
X-Pad: avoid browser bug

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
        "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
...
</body>
</html>
Connection closed by foreign host.
```

Bei Verschlüsselung: `openssl s_client -connect www.google.de:443 -crlf`

Eine Möglichkeit, Web-Server zu testen, ist `telnet` (2/2)

```
$ telnet www.informatik.hs-mannheim.de 80
Trying 141.19.145.2...
Connected to anja.ki.fh-mannheim.de.
Escape character is '^]'.
GET /~baun/test.html HTTP/1.0

HTTP/1.1 404 Not Found
Date: Sun, 04 Sep 2011 21:47:26 GMT
Server: Apache/2.2.17 (Fedora)
Content-Length: 301
Connection: close
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>404 Not Found</title>
</head><body>
<h1>Not Found</h1>
<p>The requested URL /~baun/test.html was not found on this server.</p>
<hr>
<address>Apache/2.2.17 (Fedora) Server at anja.ki.hs-mannheim.de Port 80</address>
</body></html>
Connection closed by foreign host.
```

Simple Mail Transfer Protocol (SMTP)

- Protokoll (RFC 5321) für den Austausch (Versand) von Emails
- Verwendet TCP und standardmäßig Port 25
- Das Abholen von Emails erfolgt mit den Protokollen POP3 oder IMAP
- Zum Versand von Emails verbindet sich das Mailprogramm des Benutzers mit einem SMTP-Server, der die Emails über ggf. weitere SMTP-Server zum Ziel weiterversendet
- Da SMTP ein textbasiertes Protokoll ist, kann man sich auch via Telnet mit einem SMTP-Server verbinden und so auch Emails *von Hand* versenden
  - Die Absender- und Empfängeradresse sind bei SMTP frei wählbar
    - Die Adressen im `MAIL FROM`- und `RCPT TO`-Kommando können sich von den Adressen in den Feldern `From` und `To` im Header der Email unterscheiden
  - Eine Authentifizierung findet nicht zwingend statt
    - In SMTP gibt also keine Verlässlichkeit der Absenderangabe in Emails

Statuscodes von SMTP-Servern

- Ein SMTP-Server antwortet auf Anfragen mit dreistelligen Statuscodes und kurzen Texten, die variieren oder entfallen können

Statuscode	Bedeutung	Beschreibung
`2xx`	**Erfolgreiche Ausführung**	Kommando erfolgreich ausgeführt
`4xx`	**Temporärer Fehler**	Wird das Kommando wiederholt, ist die Ausführung eventuell möglich
`5xx`	**Fataler Fehler**	Kommando kann nicht ausgeführt werden

- Die folgende Tabelle enthält einige SMTP-Kommandos

Kommando	Funktion
`HELO`	SMTP-Sitzung starten und Client identifizieren
`MAIL From:<...>`	Email-Adresse des Absenders angeben
`RCPT To:<...>`	Email-Adresse des Empfängers angeben
`DATA`	Inhalt der Email angeben
`RSET`	Eingabe einer Email abbrechen
`NOOP`	Keine Operation. Hält die Verbindung aufrecht
`QUIT`	Beim SMTP-Server abmelden

Email via SMTP mit Telnet versenden

```
$ telnet sushi.unix-ag.uni-kl.de 25
Trying 2001:638:208:ef34:0:ff:fe00:65...
Connected to sushi.unix-ag.uni-kl.de.
Escape character is '^]'.
220 sushi.unix-ag.uni-kl.de ESMTP Sendmail 8.14.3/8.14.3/Debian-5+lenny1; Mon, 5 Sep...
HELO sushi
250 sushi.unix-ag.uni-kl.de Hello sushi.unix-ag.uni-kl.de, pleased to meet you
MAIL FROM:<cray@unix-ag.uni-kl.de>
250 2.1.0 <cray@unix-ag.uni-kl.de>... Sender ok
RCPT TO:<wolkenrechnen@gmail.com>
250 2.1.5 <wolkenrechnen@gmail.com>... Recipient ok
DATA
354 Enter mail, end with "." on a line by itself
From: <cray@unix-ag.uni-kl.de>
To: <wolkenrechnen@gmail.com>
Subject: Testmail
Date: Mon, 5 Sep 2011 11:49:50 +200

Das ist eine Testmail.
.
250 2.0.0 p859lbSc018528 Message accepted for delivery
QUIT
221 2.0.0 sushi.unix-ag.uni-kl.de closing connection
Connection closed by foreign host.
```

Populäre SMTP-Server sind u.a. Exim, IBM Lotus Domino, MS Exchange, Postfix und Sendmail

Sicherheit bei SMTP

- Wegen der fehlende Sicherheitsmerkmale ist der Betrieb eines SMTP-Servers nicht ohne Sicherheitsrisiken
  - Es existiert aber Zusatzsoftware, die die Funktionalität von SMTP-Servern erweitert

Merkmal	**Bedeutung**
Zugangskontrolle	Nur zugelassene Benutzer dürfen den Server nutzen Lösungsmöglichkeiten: SMTP-Auth, SMTPS
Echtheit	Eindeutige Zuordnung zwischen Absender und Nachricht ist möglich Lösungsmöglichkeiten: PGP, S/MIME (Elektronische Unterschrift)
Integrität	Nachricht kann beim versenden nicht unbemerkt verändert werden Lösungsmöglichkeiten: PGP, S/MIME
Vertraulichkeit	Nachricht wird verschlüsselt übertragen Lösungsmöglichkeiten: PGP, S/MIME, SSL/TLS

Post Office Protocol (POP)

- Protokoll (RFC 918), das das Auflisten, Abholen und Löschen von Emails von einem Email-Server ermöglicht
- Verwendet TCP und standardmäßig Port 110
- Die aktuelle Version ist Version 3 (POP3) von 1988 (RFC 1081 und 1939)
- Die vollständige Kommunikation wird im Klartext übertragen
- Da POP3 ein textbasiertes Protokoll ist, kann man via Telnet Emails auch *von Hand* auflisten, abholen und löschen

Emails via Telnet auflisten, abholen und löschen (1/2)

Kommando	Funktion
`USER xxx`	Benutzernamen auf dem Server angeben
`PASS xxx`	Passwort angeben
`STAT`	Anzahl aller Emails im Postfach und deren Gesamtgröße (in Byte) ausgeben
`LIST (n)`	Nachrichtennummer(n) und Größe der (n-ten) Email(s) ausgeben
`RETR n`	Die n-te Email vom Server ausgeben
`DELE n`	Die n-te Email vom Server löschen
`RSET`	Alle `DELE`-Kommandos zurücksetzen
`NOOP`	Keine Operation. Hält die Verbindung aufrecht
`QUIT`	Am Server abmelden und die `DELE`-Kommandos ausführen

```
$ telnet pop.gmx.com 110
Trying 212.227.17.187...
Connected to pop.gmx.com.
Escape character is '^]'.
+OK POP server ready H migmx001
USER christianbaun@gmx.de
+OK password required for user "christianbaun@gmx.de"
PASS xyz
+OK mailbox "christianbaun@gmx.de" has 2 messages (6111 octets) H migmx107
STAT
+OK 2 6111
LIST
+OK
1 4654
2 1457
```

Emails via Telnet auflisten, abholen und löschen (2/2)

```
RETR 2
+OK
Return-Path: <wolkenrechnen@gmail.com>
Delivered-To: GMX delivery to christianbaun@gmx.de
...
From: Christian Baun <wolkenrechnen@gmail.com>
To: christianbaun@gmx.de
Subject: Testmail
Date: Mon, 5 Sep 2011 15:33:39 +0200
User-Agent: KMail/1.13.5 (Linux/2.6.35-30-generic; KDE/4.5.5; i686; ; )
MIME-Version: 1.0
Content-Type: Text/Plain;
  charset="us-ascii"
Content-Transfer-Encoding: 7bit
...

Das ist eine Testmail.
.
DELE 2
+OK
QUIT
+OK POP server signing off
Connection closed by foreign host.
```

File Transfer Protocol (FTP)

- Protokoll (RFC 959), das das Herunterladen von Dateien von FTP-Servern und Hochladen zu diesen ermöglicht
- Der vollständige Datenaustausch via FTP erfolgt im Klartext
- Sind FTP-Server und -Client miteinander verbunden, besteht zwischen beiden eine TCP-Verbindung (**Control Port**)
  - Über die Verbindung werden die Steuerkommandos zum Server gesendet
  - Dafür verwendet FTP standardmäßig Port 21
  - Auf jedes Kommando antwortet der Server mit einem Statuscode und meistens noch mit einem erklärenden Text
- Für jeden Vorgang wird jeweils eine separate TCP-Verbindung (**Data Port**) aufgebaut
  - Diese Verbindungen werden zum Senden und Empfangen von Dateien und zur Übertragung von Verzeichnislisten verwendet
  - Dafür verwendet FTP standardmäßig Port 20
  - FTP-Verbindungen können im **aktiven Modus** und im **passiven Modus** aufgebaut werden

Aktives und Passives FTP

- **Aktives FTP (*Active Mode*)**
  - Der Client öffnet einen Port mit einer Portnummer > 1023 und teilt diese und die eigene IP dem Server via `PORT`-Kommando mit
  - Der Server baut anschließend die FTP-Verbindung zwischen seinem Port 20 und dem Client-Port auf
    - **Der Verbindungsaufbau geht also vom Server aus**
- **Passives FTP (*Passive Mode*)**
  - Der Client sendet das `PASV`-Kommando an den Server
  - Der Server öffnet einen Port und sendet eine Nachricht an den Client, in der er diesem die Portnummer mitteilt
  - Der Client öffnet einen Port mit einer Portnummer > 1023 und baut die FTP-Verbindung von seinem Port zum Server-Port auf
    - **Der Verbindungsaufbau geht also vom Client aus**
  - Passives FTP wird u.a. dann verwendet, wenn der Client hinter einer Firewall liegt

Statuscodes von FTP-Servern und FTP-Kommandos

- FTP-Server beantworten jedes Kommando mit einem Statuscode, der über den Status der Kommunikation Auskunft gibt

Statuscode	Bedeutung
1xx	Kommando akzeptiert, aber noch nicht fertig ausgeführt
2xx	Kommando erfolgreich ausgeführt
3xx	Weitere Informationen sind vom Client nötig
4xx	Temporärer Fehler. Wird das Kommando wiederholt, ist die Ausführung eventuell möglich
5xx	Fataler Fehler. Kommando kann nicht ausgeführt werden

- Die folgende Tabelle enthält einige FTP-Kommandos

Kommando	Funktion
`ABOR`	Dateiübertragung abbrechen
`CWD`	Verzeichnis wechseln (*Change Working Directory*)
`DELE`	Datei löschen
`LIST`	Informationen über Datei oder Verzeichnis ausgeben
`NOOP`	Keine Operation. Hält die Verbindung aufrecht
`PASS`	Passwort angeben
`PASV`	In den Modus Passives FTP wechseln
`PORT`	IP und Portnummer für aktives FTP dem Server angeben
`PWD`	Aktuelles Verzeichnis (*Print Working Directory*)
`QUIT`	Beim FTP-Server abmelden
`SIZE`	Größe einer Datei ausgeben
`STAT`	Status der Verbindung angeben
`USER`	Benutzernamen angeben

FTP-Server via Telnet bedienen (1/2)

- Da FTP ein textbasiertes Protokoll ist, kann man auch via Telnet mit FTP-Servern arbeiten

```
$ telnet ftp.kernel.org 21
Trying 130.239.17.5...
Connected to pub.eu.kernel.org.
Escape character is '^]'.
220 Welcome to ftp.kernel.org.
USER anonymous
331 Please specify the password.
PASS guest
230-                        Welcome to the
230-
230-                    LINUX KERNEL ARCHIVES
230-                        ftp.kernel.org
230-
230-                "Much more than just kernels"
...
230 Login successful.
PASV
227 Entering Passive Mode (199,6,1,165,95,157).
PWD
257 "/"
NOOP
200 NOOP ok.
```

FTP-Server via Telnet bedienen (2/2)

```
STAT
211-FTP server status:
     Connected to 84.171.167.112
     Logged in as ftp
     TYPE: ASCII
     No session bandwidth limit
     Session timeout in seconds is 300
     Control connection is plain text
     Data connections will be plain text
     At session startup, client count was 56
     vsFTPd 2.3.4 - secure, fast, stable
211 End of status
CWD /pub
250 Directory successfully changed.
PWD
257 "/pub"
SIZE README
213 1912
SIZE index.html
213 2322
QUIT
221 Goodbye.
Connection closed by foreign host.
```

- Eine verschlüsselte Alternative zu FTP ist das Secure File Transfer Protocol (SFTP)