Hitchhiker´s Guide to the Internet

Ingo Blechschmidt

6. Juni 2005

Inhaltsverzeichnis

1 Einleitung **6**

1.1 Bezugsquellen 6

1.2 Schreibkonventionen 6

1.3 Wishlist 7

2 Grundlagen **8**

2.1 Das OSI-Schichtenmodell 8

2.2 IP 9

2.3 Routing 9

2.3.1 Time to Live 11

2.3.2 Traceroute 11

2.3.3 OS-Fingerprinting mittels der TTL 13

2.4 TCP 15

2.4.1 Telnet 15

2.4.2 nmap 16

2.5 UDP 17

2.5.1 Netcat 18

2.6 ICMP 18

2.6.1 Ping 19

2.7 Abschluss 20

3 E-Mails **20**
3.1 Format 21
3.2 Typische Header 21
3.3 SMTP-Server-Stempel 22
3.4 MIME-Typen 23
3.5 Abschluss 24

4 SMTP **24**
4.1 Grundlagen 24
4.2 Envelope-Header 25
4.3 Typischer Ablauf 26
4.4 Beispiel 29
4.5 Befehlsübersicht 31

5 POP3 **31**
5.1 Grundlagen 31
5.2 Typische POP3-Sitzung 31
5.3 Protokollablauf 35
5.4 Befehlsübersicht 37

6 IMAP4 **37**
6.1 Technische Implementierung 37
6.2 Befehlsübersicht 40
6.3 Abschluss 40

7 NNTP **41**
7.1 Design 42
7.2 Beispielsitzung 42
7.2.1 Zusammenfassung 45
7.3 Besondere Header 47
7.3.1 »Newsgroups« 47

7.3.2 »Followup-To« 47
7.3.3 »Path« 47
7.3.4 »Approved« 48
7.3.5 »Control« 48
7.4 Austausch zwischen den Servern 48
7.5 Befehlsübersicht 50
7.6 Abschluss 50

8 HTTP 52

8.1 Versionen 52
8.2 Adressen 52
8.3 HTTP/1.0 53
8.4 HTTP/1.1 54
8.5 HTTP-Proxies 55
8.6 Überblick 55
8.7 Referer 56
8.8 Weiterleitung 56
8.9 HEAD-Request 57
8.10 TRACE-Request 58
8.11 User-Agent 58
8.12 Accept-Language 59
8.13 Keep-Alive 59
8.13.1 Chunked-Encoding 60
8.14 Partial Content 61
8.15 Cookies 62
8.16 Formulare 63
8.16.1 GET-Request 64
8.16.2 POST-Request 65
8.17 Requestübersicht 66
8.17.1 Beispielsitzung 66

9 Gopher **67**
9.1 Design 68
9.2 Technik 68
9.3 Beispielsitzung 68
9.4 Abschluss 70

10 FTP **70**
10.1 Design 71
10.2 Passives FTP 71
10.3 Aktives FTP 74
10.4 Befehlsübersicht 76
10.5 Probleme mit Firewalls 76
10.6 Goodies 77
10.6.1 Übertragung zwischen zwei Servern 77
10.7 Application-Level-Proxy 78
10.8 Abschluss 79

11 IRC **79**
11.1 Design 80
11.2 Beispielsitzung 80
11.3 Befehlsübersicht 84
11.3.1 Nachrichtenversand 84
11.3.2 Benutzerinformationen 85
11.4 Modes 87
11.4.1 Benutzermodes 88
11.4.2 Channelmodes 90
11.5 CTCP 91
11.6 DCC 92
11.6.1 XDCC 95

12 DICT **95**
12.1 Design 96
12.2 Beispielsitzung 96
12.3 Befehlsübersicht 98
12.4 Anwendungen 99

13 Finger **99**
13.1 Design 99
13.2 Beispiel 100
13.3 Anwendungen 101

14 Ident **101**
14.1 Technische Realisierung 102
14.2 Anwendungen 103

15 Daytime **103**
15.1 Beispiel 104

16 DNS **104**
16.1 Geschichte 104
16.2 Design 105
16.3 Vor- und Nachteile 107
16.4 Record-Typen 107
16.5 Ausfallsicherung 108
16.6 MX-Records 111
16.7 Reverse-Lookups 112
16.8 Whois 112

17 Muhahaha – oder: Automatisierung **114**
17.1 Whois 114
17.2 Daytime 115

17.3 DICT 116
17.4 HTTP 117
17.4.1 Lynx 117
17.4.2 curl 119
17.4.3 Standardsyntax 120
17.4.4 Perl-Modul HTTP::Recorder 121

1 Einleitung

In diesem Dokument werden die verschiedenen Internetprotokolle (sowohl Low-Level als auch Application-Level) vorgestellt, so dass man nach Studieren dieser Datei (welche man im Übrigen auch in einem beliebigen ASCII-Viewer betrachten kann) in der Lage ist, nur via Telnet zu surfen, Mails zu verschicken und empfangen, usw.

Geschrieben wurde diese Version (2787) mit MyBook (programmiert vom Autor).

Über jegliche Kommentare und Ergänzungen würde ich mich, Ingo Blechschmidt, »iblech@web.de«, sehr freuen.

1.1 Bezugsquellen

Die aktuellste Version dieses Buches ist auf »http://linide.sf.net/theguide2/«[1] zu finden. Interessant könnte auch die Projektseite auf Freshmeat[2] sein.

1.2 Schreibkonventionen

Einzelne Befehle oder Programme werden im Fließtext »gesperrt« gedruckt.

Längere Listings hingegen bekommen immer einen extra Absatz:

---

[1] http://linide.sf.net/theguide2/
[2] http://freshmeat.net/projects/theguide/

```
Dies ist die erste Zeile. \
    Dies ist immer noch die erste Zeile, musste aber \
    aus Platzgründen mit einem "\" umgebrochen werden.
```

Ein Dialog, etwa zwischen Server und Client, wird wiefolgt dargestellt:

```
Client (Anfrage)
Server (Antwort)
```

Werden im Fließtext Programme genannt, werden sie »gesperrt« gedruckt (Beispiel: »telnet« erwartet als Parameter...). Wenn aber das Konzept oder die Idee hinter dem Programm gemeint ist, wird es normal gedruckt und passt sich auch der Groß-/Kleinschreibung an (Beispiel: Die Idee hinter Telnet...).

Aus Platzgründen müssen in Listings oft einige Details weggelassen werden.

1.3 Wishlist

Ich kann auch noch andere Protokolle hier erläutern, einfach mir eine Mail schicken und ich arbeite daran. . .

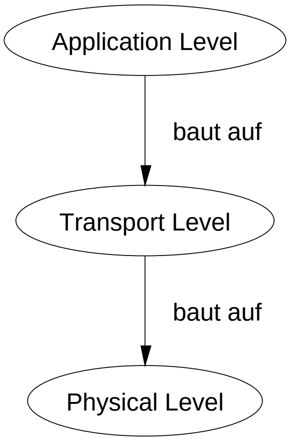
- SMTP: Spam-Abwehrmaßnahmen
- Was sind RFCs?
- HTTP: Transfer-Encoding gzip?
- Web-Services
- IPv6
- DICT-Automatisierung
- EPOP3?
- Mehr Protokolle, aber welche. . . (=bitte Mail an mich)?

2 Grundlagen

In diesem Kapitel werden die unteren Schichten des Internets, also die Basis, auf der alle folgenden Kapitel aufbauen werden, vermittelt.

2.1 Das OSI-Schichtenmodell

Man untergliedert die Protokolle des Internets in verschiedene Schichten, definiert schon 1982 durch das OSI-(„Open System Interconnection“)-Schichtenmodell. Vereinfacht dargestellt, gliedert es alle Protokolle in drei Schichten[3]:



Je weiter „oben“ ein Protokoll liegt, so abstrakter ist es. Mit diesen Protokollen werden wir uns am meisten beschäftigen. „Unten“ liegt zum Beispiel die physikalische Schicht: Das entspräche praktisch dem Netzwerkkabel. Da uns aber nur die Software interessiert, wird hier darauf nicht eingegangen werden.

Auf der Transportschicht ist das Protokoll IP, „Internet Protocol“, angesiedelt. IP ist für die grundlegende Kommunikation aller Rechner im Internet zuständig. Eine Schicht höher (nicht abgebildet) liegt TCP, das Transport Control Protocol. Dieses Protokoll sorgt dafür, dass die mit IP versendeten Pakete am Ziel auch ankommen, da IP selbst nicht für die Lieferung der Pakete garantiert. Auf

[3]In Wirklichkeit sind es sieben Schichten, nachzulesen im Wikipedia-Eintrag zum Thema[4].

der Anwendungsebene schließlich sind alle „hohen“ Protokolle angesiedelt, namentlich SMTP, POP3, HTTP, NNTP und viele andere, denen je ein einzelnes Kapitel gewidment ist.

2.2 IP

Heute sorgt für die Kommunikationsfähigkeit aller Knoten des Internets die Version 4 des IP.

Sendedaten unterteilt IP in kleinere Pakete, die dann verschickt werden können. Dabei werden Absender und Empfänger durch eine IP-Adresse bestimmt.

Alle IP-Adressen sind vier Byte lang (ein „Word“), untergliedert in vier Zahlen mit je einem Byte. Was sich hier etwas kompliziert anhört ist ganz einfach: »80.81.9.177« ist zum Beispiel eine gültige IP-Adresse. Jede Zahl darf (wegen der 1-Byte-pro-Zahl-Grenze) maximal 255 (die 0 wird mit einbezogen) betragen.

Besondere IP-Adressen sind solche, die

- auf ».0« enden. Diese IP-Adressen definieren ein ganzes Subnetz. So definiert »10.0.0.0« zum Beispiel ein Subnetz, mit dem alle Rechner, dessen IP-Adressen mit »10.« beginnen, gemeint sind. Solche IP-Adressen können also **nicht** dazu verwendet werden, um einen einzelnen Rechner anzusprechen.
- Außerdem haben IP-Adressen, die auf ».255« enden, sogenannte „Broadcast“-Adressen, eine besondere Bedeutung:

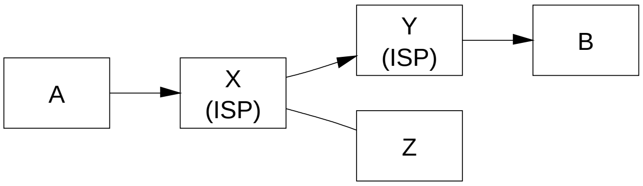
  Jeder Rechner eines Subnetzes hört auf Pakete, die an diese IP-Adresse geschickt wurden. Schickt man zum Beispiel einen Ping (auf Seite 19) an eine solche Adresse, „pongen“ alle anderen Rechner des Subnetzes zurück.

2.3 Routing

Aber wie kommen die Pakete von einem Rechner »A« zu »B«? Da gäbe es jetzt zwei Möglichkeiten:

- Man verbindet jeden Rechner des Internets mit jedem anderen. Schlecht.

- Man leitet die Pakete über andere Rechner, die dann als **Router** fungieren, d.h., sie leiten die Pakete nur weiter, verarbeiten sie aber nicht.



Dabei hat jeder Rechner in seiner **Routing-Tabelle** gespeichert, welche Verbindung zu welchem Rechner führt.

Im Beispiel gilt:

»**A**«

- Wenn an »X«, direkte Leitung benutzen.
- Alle anderen Pakete an »X«, den Internet Service Provider (ISP), weiterleiten.

»**X**«

- Wenn an »A«, direkte Leitung benutzen.
- Wenn an »Z«, direkte Leitung benutzen.
- Wenn an »Y«, direkte Leitung benutzen.
- Alle anderen Pakete an »Y« weiterleiten.

»**Y**«

- Wenn an »B«, direkte Leitung benutzen.
- Wenn an »X«, direkte Leitung benutzen.
- Alle anderen Pakete an »X« weiterleiten.

»**Z**«

- Wenn an »X«, direkte Leitung benutzen.
- Alle andere Pakete an »X« weiterleiten.

»**B**«

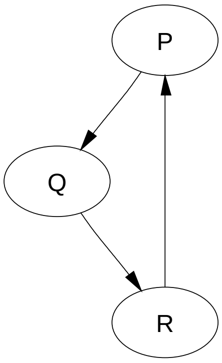
- Wenn an »Y«, direkte Leitung benutzen.
- Alle andere Pakete an »Y«, den ISP, weiterleiten.

2.3.1 Time to Live

Durch z.B. fehlerhafte Routing-Tabellen können nun aber auch Endlosschleifen entstehen: Meint z.B. ein Rechner »P«, er müsse alle Pakete an »Q« schicken, und »Q« meint, alle Pakete sollen an »P« weitergeleitet werden, so wird das Paeket ewig zwischen den beiden Rechnern weitergeleitet werden.

Eine naive Lösung dieses Problems wäre es, einfach festzulegen: „Schicke niemals ein Paket zu dem Rechner zurück, der es dir zugeschickt hat.“

Aber auch dadurch wird das Problem nicht vollständig gelöst, nämlich dann, wenn noch ein weiterer Rechner „in der Schleife festsitzt“:



Deswegen kommt hier ein intelligenteres Verfahren zum Zug: Jedes Paket gibt in seinem Header nicht nur über Absender und Empfänger auskunft, sondern auch über die sogenannte **Time to Live**, abgekürzt TTL. Die TTL ist eine ein Byte breite Zahl (mögliche Werte zwischen »0« und »255«). Wenn das Paket beim Absender generiert wird, erhält es einen bestimmten Startwert, der sich bei den meisten Betriebssystemen voneinander unterscheidet. Jedesmal, wenn das Paket einen Router passiert, wird dieser Wert um eins dekrementiert. Ist die TTL »0«, wird das Paket verworfen und an den Absender wird eine Fehlermeldung über das Protokoll ICMP (auf Seite 18) zugestellt.

2.3.2 Traceroute

Mit Hilfe der TTL kann man auch herausfinden, wie viele Router ein Paket passieren musste, ehe es sein Ziel erreichte. Dabei sendet man zuerst ein Paket zum Empfänger mit einer Start-TTL von »1«. Erhält man keine Fehlermeldung, so besteht eine direkte Leitung zum Empfänger. Andernfalls war die TTL zu niedrig und man sendet erneut ein Paket, aber diesmal mit einer TTL von »2«. Dieses Spiel führt man so lange fort, bis man bis zum Ziel durchkommt.

Möchte man diesen Vorgang automatisieren, benutzt man den Shellbefehl »traceroute« (oder »tracert« auf schlechten Betriebssystemen), den man als root ausführen muss:

```
thestars theguide # traceroute irc.lugs.ch
traceroute to wigwam.ethz.ch, 30 hops max
 1  mars (10.0.0.4)
 2  ascend7.augustakom.net (80.81.6.71)
 3  router1.augustakom.net (80.81.6.2)
 4  80.81.7.118 (80.81.7.118)
 5  A.S-3-eth000-106.de.lambdanet.net (217.71.108.37)
 6  F-2-pos030-0.de.lambdanet.net (217.71.105.117)
 7  80.86.163.22 (80.86.163.22)
 8  swiix1-g2-1.switch.ch (194.42.48.11)
 9  swiEZ2-G3-2.switch.ch (130.59.36.249)
10  rou-rz-gw-giga-to-switch.ethz.ch (192.33.92.1)
11  rou-ethz-access-intern.ethz.ch (192.33.92.130)
12  rou-hpx-1-mega-transit-2.ethz.ch (129.132.99.199)
13  wigwam.ethz.ch (129.132.189.109)
thestars theguide #
```

Traceroute ist oft als Diagnoseprogramm sinnvoll, wenn man eine Fehlermeldung der Art „Time to Live exceeded" zu Gesicht bekommt[5]. Sind einige Router „leicht fehlerhaft" konfiguriert, kann die Ausgabe z.B. so aussehen:

```
16  router1.augustakom.net (80.81.6.2)
17  router2.augustakom.net (80.81.7.118)
18  router1.augustakom.net (80.81.6.2)
19  router2.augustakom.net (80.81.7.118)
20  router1.augustakom.net (80.81.6.2)
21  router2.augustakom.net (80.81.7.118)
(...)
```

[5] Das ist mir in der Tat einmal passiert: Die Router meines ISPs spielten Ping-Pong mit meinen Paketen, Router »A« schickte sie zu Router »B«, »B« zu »A«, usw. (Hallo estel ;-))

»traceroute« verschickt standardmäßig UDP-Pakete (auf Seite 17), kann aber auch mit der Option »-I« ICMP-Echo-Request-Pakete (auf Seite 19) verschicken. Mit dem exzellenten »tcptraceroute« von Michael Toren[6] können auch TCP-Pakete (auf Seite 15) verschickt werden.

2.3.3 OS-Fingerprinting mittels der TTL

Wie weiter oben schon kurz angesprochen nehmen viele Betriebssysteme einen anderen Startwert für die TTL her. Umgekehrt bedeutet dies: Kennt man die Start-TTL eines Paketes, kann man auch mit einiger Gewissheit sagen, welches Betriebssystem der Absender benutzt. Dieser Vorgang ist ein Teil des sogenannten „**OS-Fingerprintings**“, dem Identifizieren des eingesetzten Betriebssystems (und evtl. seiner Version).

Bei der praktischen Umsetzung dieser Idee gibt es jedoch noch ein Problem: Über (z.B.) einen Ping (auf Seite 19) erfährt man nur den Wert der TTL am Ende der Reise des Pakets, der Startwert bleibt unbekannt. Aber dank Traceroute kann man ja auch die Anzahl der Hops, die Anzahl der Router, die Pakete auf dem Weg zum Ziel passieren mussten, bestimmen. Addiert man nun also die TTL des Paketes am Ende seiner Reise und die Anzahl der Hops, so erhält man die Start-TTL. Diese kann man dann in Tabellen nachschlagen.

Als Beispiel vergleichen wir die Start-TTLs von »www.debian.de« und »www.suse.de«. Wir gehen davon aus, dass sie beide eine ähnliche Version von Linux installiert haben, also müssten ihre Start-TTLs miteinander übereinstimmen.

- Zuerst bestimmen wir die TTL, die „übrig bleibt“, sobald wir »www.debian.de« erreichen:

```
thestars theguide # ping -c1 www.debian.de
PING www.de.debian.org (141.76.2.5)
64 bytes from 141.76.2.5: ttl=50 time=46.5 ms

--- www.de.debian.org ping statistics ---
1 packets transmitted, 1 received, 0% packet loss
rtt min/avg/max/mdev = 46.528/46.528/46.528/0.000 ms
thestars theguide #
```

[6] http://michael.toren.net/code/tcptraceroute/

Die TTL beträgt also »50«.

- Nun bestimmen wir die Anzahl der Hops, die zwischen uns und »www.debian.de« liegen:

```
thestars theguide # traceroute www.debian.de
traceroute to www.de.debian.org (141.76.2.5)
 1  mars (10.0.0.4)
 2  router0.augustakom.net (80.81.6.1)
 3  router1.augustakom.net (80.81.6.2)
(...)
14  cat6k-inf.campus.urz.tu-dresden.de (141.30.1.114)
15  www.de.debian.org (141.76.2.5)
thestars theguide #
```

- Nun müssen wir noch die Start-TTL errechnen. Dabei ist es wichtig, dass wir für die Anzahl der Hops nicht **15**, sonden **14** nehmen: Das Ziel selbst, der 15. Rechner, den uns »traceroute« angezeigt hat, dekrementiert die TTL ja nicht.

```
thestars theguide # echo 50 + 14 | bc
64
thestars theguide #
```

Der Startwert der TTL von Paketen, die »www.debian.de« versendet, ist also »64«.

- Jetzt wiederholen wir den Vorgang mit »www.suse.de«.

Pingen. . .

```
thestars theguide # ping -c1 wwww.suse.de
PING turing.suse.de (195.135.220.3)
64 bytes from 195.135.220.3: ttl=54 time=42.5 ms

--- turing.suse.de ping statistics ---
1 packets transmitted, 1 received, 0% packet loss
rtt min/avg/max/mdev = 42.558/42.558/42.558/0.000 ms
thestars theguide #
```

- ...die Anzahl der Hops ermitteln. . .

```
thestars theguide # traceroute www.suse.de
traceroute to turing.suse.de (195.135.220.3)
 1  mars (10.0.0.4)
 2  router0.augustakom.net (80.81.6.1)
 3  router1.augustakom.net (80.81.6.2)
(...)
10  * * *
11  skylla-router.suse.de (195.135.221.1)
thestars theguide #
```

- ...und zusammenzählen:

```
thestars theguide # echo 54 + 10 | bc
64
thestars theguide #
```

Die Start-TTLs stimmen miteinander überein, unsere Vermutung, dass »www.suse.de« und »www.debian.de« das gleiche Betriebssystem einsetzen, war also korrekt. Wir könnten jetzt auch noch in Tabellen nachschlagen, welches Betriebssystem normalerweise Pakete mit einer TTL von »64« sendet, aber im Fall von Debian und SuSE ist das ziemlich klar. . . ;-).

2.4 TCP

Ein gravierender Nachteil von IP ist allerdings die mangelnde Fehlertoleranz: Ist Netzlast hoch, kommen viele Pakete nicht am Ziel an. Deswegen wurde ein weiteres Protokoll entworfen, TCP, das Transmission Control Protocol. TCP sorgt dafür, dass die via „normalem“ IP versendeten Pakete auch wirklich am Ziel ankommen.

Dies erreicht TCP vereinfacht gesagt dadurch, dass es die Pakete nummeriert. Empfängt der Zielrechner z.B. die Pakete mit den Nummern 42, 43, 45 und 46, so weiß er, dass Paket 44 fehlt und kann es neu anfordern.

Auch ergänzt TCP IP um sogenannte Ports: Auf jedem der insgesammt 2^{16} Ports (»0« bis »65535«[7]) kann ein eigener Dienst (HTTP, SMTP, POP3, IMAP, DNS, etc.) „lauschen“. Dadurch erst wird die Dienstevielfalt des Internets möglich.

2.4.1 Telnet

Um zu einem TCP-Port eines Hosts zu connecten, benutzt man unter guten System (Linux, Hurd) den Shellbefehl »telnet«. Um z.B. eine Verbindung mit dem Rechner »mars« auf Port »22« herzustellen, benutzt man:

[7] Obwohl der Port »0« theoretisch verwendet werden könnte, hat er in der Praxis kaum eine Bedeutung, da auf unixoiden Systemen der Port »0« verwendet wird, um das Betriebssystem einen freien Port auswählen zu lassen.

```
iblech@thestars theguide $ telnet mars 22
Trying 10.0.0.4...
Connected to mars.gnus.
Escape character is '^]'.
SSH-2.0-OpenSSH_3.8.1p1
^] (Strg+AltGr+] wird auf diese Weise angezeigt)
telnet> q
Connection closed.
iblech@thestars theguide $
```

Um eine Verbindung vorzeitig abzubrechen, kann man die Tastenkombination »Strg«+»AltGr«+»]« benutzen. Daraufhin nimmt »telnet« Befehle entgegen. Mit »quit« (abkürzbar auf »q«) kann man die Verbindung schließen.

„Aber Telnet ist doch unsicher!1 Telnet sollte nicht verwendet werden!"

Diese Aussage, für die Google immerhin über 3000 Ergebnisse[8] liefert, ist nur bedingt richtig. Richtig ist, dass bei Telnet alle Daten im Klartext, also unverschlüsselt, übertragen werden. So kann, durch Abhören des Netzverkehrs („Sniffen"), auch sensible Daten wie Passwörter mitgeschnitten werden. Möchte man Telnet also zur Fernadministration einsetzen, ist diese Aussage zweifellos richtig und man sollte lieber OpenSSH[9] einsetzen. SSH verschlüsselt den Datenstrom bevor er über das Netz gesendet wird.

Aber bei allen anderen Einsatzgebieten von Telnet kann man nicht pauschal von einer Unsicherheit reden. Möchte man z.B. nur die aktuelle Zeit abfragen[10], steht die Sicherheit[11] im Hintergrund.

2.4.2 nmap

Möchte man eine Übersicht aller offenen Ports (Ports, an denen ein Dienst lauscht), verwendet man einen Portscanner.

[8] http://www.google.com/search?&q=telnet%20unsicher

[9] http://www.openssh.org/

[10] Es gibt viele Zeitserver im Internet, die u.A. auf Port »13« über das Daytime-Protokoll (mehr dazu im Kapitel über Daytime (auf Seite 103)) oder über andere Protokolle die aktuelle Zeit liefern. Zum Zeitpunkt des Schreibens bietet ein Rechner der TU-Berlin (»sombrero.cs.tu-berlin.de«) diesen Service an: Mit einer Verbindung auf Port »13« antwortet er mit der aktuellen Zeit in einem Menschen-lesbaren Format.

[11] meistens :-)

Portscanner verbinden sich praktisch mit jedem möglichen Port des Zielsystems. Wird die Verbindung aufgebaut, ist der Port offen und wird angezeigt. Ein beliebter Portscanner unter Linux und anderen Unix-basierten Systemen ist nmap. »nmap« erwartet in seiner einfachsten Form nur die Namen der Hosts, die gescannt werden sollen:

```
iblech@thestars theguide $ nmap thestars

Starting nmap 3.50 ( http://www.insecure.org/nmap/ )
Interesting ports on thestars.gnus (10.0.0.3):
(The 1651 ports scanned but not shown are: closed)
PORT     STATE SERVICE
22/tcp   open  ssh
23/tcp   open  telnet
53/tcp   open  domain
79/tcp   open  finger
1024/tcp open  kdm
6000/tcp open  X11
6666/tcp open  irc-serv
6667/tcp open  irc

Nmap run completed -- 1 IP address (1 host up) scanned
iblech@thestars theguide $
```

Möchte man schon während dem Scan sehen, welche Ports als offen identifiziert wurden, kann man »nmap« mit der Option »-v« aufrufen:

```
iblech@thestars theguide $ nmap -v thestars
```

2.5 UDP

UDP, das User Datagram Protocol, ergänzt IP lediglich um die schon von TCP bekannten Ports, nicht aber um die Fehlertoleranz. Pakete, die aus irgendeinem Grund nicht am Ziel ankommen, werden also **nicht** nochmal geschickt.

Dies ist z.B. bei der Übertragung von Audio- und Video-Streams sinnvoll, da dort eine evtl. häufige Neu-Übertragung von Paketen die verfügbare Bandbreite nur unnötig schmälern würde. Außerdem fallen einige nicht übertragene Pakete nicht ins Gewicht: Das nächste Paket, welches z.B. die nächste zehntel Sekunde eines Audio- oder Videostreams beschreibt, wird schon nach sehr kurzer Zeit abgesendet. Das Fehlen einen Frames wird quasi durch den nächsten „übertönt“, es ist höchstens ein kurzes Knacken zu hören bzw. ein kurzer Hänger zu sehen.

Auch gibt es bei UDP nicht das Konzept einer Verbindung zwischen zwei Hosts: Es werden einfach Pakete verschickt und empfangen, aber es gibt keine Zugehörigkeit zu einer Verbindung. Ein Server kann auf ein UDP-Paket in dem Sinne auch nicht antworten, sondern schickt einfach ein neues Paket los.

2.5.1 Netcat

Möchte man manuell eine „Verbindung“ zu einem UDP-Port herstellen, kann man Netcat verwenden, Telnet ist dazu nicht fähig.

Als Beispiel wollen wir ein Paket zum UDP-Port »13« von »sombrero.cs.tu-berlin.de« schicken:

```
iblech@thestars theguide $ nc -u sombrero.cs.tu-berlin.de
13
we be leet
Tue Aug 10 14:04:49 2004
(Strg+C)
iblech@thestars theguide $
```

Statt »we be leet« hätten wir auch nur eine Leerzeile oder etwas anderes schicken können: Der Server von »sombrero.cs.tu-berlin.de«, der auf dem UDP-Port »13« lauscht, ist so programmiert, dass er, immer, wenn er ein Paket empfängt, er ein Paket mit der aktuellen Zeit zurückschickt. Auch mussten wir »nc« mit »Strg«+»C« abbrechen: Da es bei UDP ja keine Verbindungen gibt, konnte der Server auch keine schließen, was für Netcat das Signal gewesen wäre, sich zu beenden. Aber dies ist UDP, nicht TCP, also mussten wir selbst das Programm beenden.

2.6 ICMP

ICMP, das Internet Control Message Protocol, wird, von einigen kryptographischen Zwecken einmal abgesehen, nur zur Statusübertragung für IP/TCP/UDP eingesetzt, zum Beispiel ob der Zielhost existiert, eine Route verfügbar ist usw. Dies haben wir weiter oben bei Traceroute (auf Seite 11) ausgenutzt: Der Fehler, der Traceroute gemeldet wird (Time-to-Live exceeded), wird über ICMP übertragen.

Wichtig dabei ist, dass Fehler, die bei der Übertragung der Statusmeldungen auftreten, **nicht** nochmal gemeldet werden. Unter Umständen wäre eine endlose Überschwemmung des Netzes die Folge.

2.6.1 Ping

Von den verschiedenen Statusmeldungen, die über ICMP übertragen werden können, werden zwei besonders häufig genutzt: „Pings" und „Pongs" (ICMP-Echo-Requests und ICMP-Echo-Replies).

Erhält ein Rechner einen Ping, so sollte er einen Pong zurückschicken. Dies wird oft genutzt, um die Erreichbarkeit von Hosts zu testen.

Das Programm, das Pings versendet, heißt unerwarteterweise »ping«. Der Aufruf ist simpel, als Argument erwartet »ping« lediglich den Host, den es pingen soll:

```
iblech@thestars theguide $ ping mars
PING mars.gnus (10.0.0.4) 56(84) bytes of data.
64 bytes from mars: icmp_seq=1 ttl=64 time=1.02 ms
64 bytes from mars: icmp_seq=2 ttl=64 time=0.208 ms
64 bytes from mars: icmp_seq=3 ttl=64 time=0.203 ms
(Strg+C)
--- mars.gnus ping statistics ---
3 packets sent, 3 received, 0% packet loss, time 2011ms
rtt min/avg/max/mdev = 0.203/0.478/1.024/0.386 ms
iblech@thestars theguide $
```

Auf guten System pingt »ping« bis es manuell abgebrochen wird, auf schlechten Systemen (=MDollar) wird nur einige Male gepingt.

Ist ein Rechner nicht online, erhält man eine Ausgabe der Art

```
iblech@thestars theguide $ ping trinity
PING trinity.gnus (10.0.0.6) 56(84) bytes of data.
(Strg+C)
--- trinity.gnus ping statistics ---
2 packets sent, 0 received, 100% packet loss

iblech@thestars theguide $
```

„Pings sind böse! Auf Pings darf man nicht antworten! Sonst weiß ein böser Hacker, dass man online ist!“

In der Tat kann man „Pings einfach blocken“, also keine Antwort auf Pings verschicken. Aber sicherer vor „bösen Hackern“ ist man deswegen nicht: Zum einen reagieren die meisten Provider auf einen Ping auf einen Rechner, der offline ist, mit einem ICMP-Destination-Unreachable-Paket. Konfiguriert man nun seine Firewall so, dass keine Pongs verschickt werden, weiß ein Angreifer, dass der online ist: Wäre er es nicht, würde vom Provider ja das besagte ICMP-Destination-Unreachable-Paket kommen.

Zum anderen gibt es noch andere Methoden, um festzustellen, ob ein Rechner online ist: Zum Beispiel könnte man einfach zu irgendeinem (TCP-)Port des Rechners connecten. Schlägt der Versuch sofort fehl (nicht erst nachdem ein Timeout abgelaufen ist), ist klar, dass der Rechner on ist: Sonst hätte er den Versuch eines Verbindungsaufbau ja nicht zurückweisen können.

Besser ist es, die Dienste, die man anbietet, abzusichern, und nicht „Pings zu blocken“ und darauf zu hoffen, dass man dann sicher ist.

2.7 Abschluss

Besonders den Umgang mit »telnet« sollte man beherrschen, um die Beispiele der nächsten Kapitel auch selbst ausprobieren zu können.

3 E-Mails

Oft wird dazu geraten, Spam-Mails („Unerwünschte Werbe-Mails“) einer sogenannten **Header-Analyse** zu unterziehen.

Dabei ist der Ausdruck insbesondere beim Umgang mit Mails nicht ganz korrekt: Bei E-Mails gibt es zwei Header-Typen, einmal den SMTP-„Header“ (mehr dazu im Kapitel über SMTP (auf Seite 24)) und dann den Mail-Header, um den es in diesem Kapitel geht.

3.1 Format

Schaut man sich einmal den vollständigen Quelltext einer Mail an, so stellt man fest, dass die Mail in zwei Abschnitte gegliedert ist.

Zuerst kommt der Header, der Auskunft über Absender, Empfänger, etc. gibt. Die einzelnen Headerfelder werden von den Feldinhalten mit einem »: « (Doppelpunkt Leerzeichen) voneinander getrennt.

Dann folgt, getrennt durch eine Leerzeile, der Nachrichteninhalt. Eine optionale Signatur wird durch »-- « (Bindestrich Bindestrich Leerzeichen) vom Text getrennt.

```
Received: from imap.web.de [217.72.192.135]
    by localhost with IMAP (fetchmail-6.2.5)
    for iblech@localhost (single-drop);
    Thu, 05 Aug 2004 18:55:15 +0200 (CEST)
Received: (nullmailer pid 9702 invoked by uid 1000);
    Thu, 05 Aug 2004 16:54:02 -0000
Date: Thu, 5 Aug 2004 18:54:02 +0200
From: Ingo Blechschmidt <iblech@web.de>
To: Ingo Blechschmidt <iblech@thestars.gnus>
Subject: Test
Message-ID: <20040805165402.GA9665@thestars.gnus>
Reply-To: iblech@web.de
Content-Type: text/plain; charset=us-ascii
User-Agent: Mutt/1.5.6i
Content-Length: 190
Lines: 7

Hier da

--  (Bindestrich Bindestrich Leerzeichen)
Linux, the choice of a GNU | Mathematicians practice
generation on a dual AMD-  | absolute freedom.
Athlon!                    |     -- Henry Adams
Encrypted mails preferred. |
```

3.2 Typische Header

Die meisten Header sind in RFC 2076[12] standardisiert. Die mit »X-« beginnenden Header können frei verwendet werden, auch wenn

[12] http://www.faqs.org/rfcs/rfc2076.html

sich einige X-Header auch schon als de-facto Standard durchgesetzt haben:

Header	Bedeutung
»Received«	SMTP-Server-Stempel
»Date«	Zeitpunkt des Versendens
»From«	Absender[13]
»Reply-To«	Adresse, an die Rückantworten gehen sollen
»Message-ID«	Weltweit eindeutige ID, oft bestehend aus einem festen und einem zufälligen Teil. Die Eindeutigkeit der Message-ID ist besonders bei NNTP (auf Seite 41) sehr wichtig.
»In-Reply-To«	Message-ID, auf die geantwortet wurde
»To«, »Cc«	Empfänger
»Subject«	Betreff
»Organization«[14]	Organisation, Firma, etc.
»Content-Type«	MIME-Typ (auf der nächsten Seite)
»User-Agent«, »X-Mailer«	Verwendeter MUA (Mailprogramm)
»X-Operating-System«	Zum Verfassen der Mail verwendetes Betriebssystem (Linux, Hurd, ...)
»X-GnuPG-Key«	ID des öffentlichen GnuPG/PGP-Schlüssels

3.3 SMTP-Server-Stempel

Mails werden, wie im nächsten Kapitel (auf Seite 24) beschrieben, bei SMTP über viele Server geleitet. Jeder Server fügt beim Weiterleiten seinen eigenen »Received«-Header möglichst weit unten (aber vor den „normalen" Headern wie »From« usw.) an, zum Beispiel:

[13] Diese Absenderadresse kann extrem leicht gefälscht werden, wie das Kapitel über SMTP noch zeigen wird. . .

[14] Oft auch mit »s« statt »z« geschrieben

```
Received: from imap.web.de [217.72.192.135]
    by localhost with IMAP (fetchmail-6.2.5)
    for iblech@localhost (single-drop);
    Thu, 05 Aug 2004 18:55:15 +0200 (CEST)
```

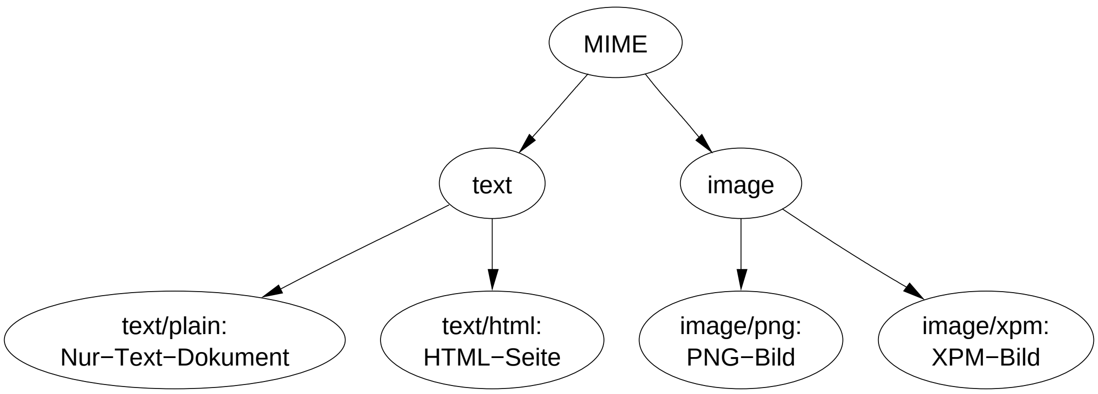
Hier hat also der Server »localhost« vom Server »iblech.dyndns.org« eine Mail via dem Protokoll ESMTP erhalten.

So lassen sich manchmal Spam-Versender zurückverfolgen, auch wenn sie ihre Absenderadresse fälschen: Es ist ein leichtes, den »From«-Header zu fälschen. Aber der (nicht so einfach modifizierbare) erste »Received«-Header lässt den der Ursprung der Mail trotzdem erkennen.

3.4 MIME-Typen

Ein anderer wichtiger Header ist »Content-Type«. Dieses Feld gibt den MIME-Typ („Multipurpose Internet Mail Extensions“) des Inhalts der Mail an. Nur so weiß der Mailreader, welches Format die Mail hat („Enthält die Mail ausschließlich Text? Oder ist sie vielleicht in HTML verfasst?“).

MIME-Typen sind untergliedert in Kategorien:



MIME-Typen werden heute auch bei HTTP eingesetzt (auf Seite 54), damit der Browser auch ohne die eher von altmodischen Betriebssystemen (sic) bekannte Dateinamenserweiterung über den Typ einer Datei bescheid weiß.

3.5 Abschluss

Über den Header kann man also einige wichtige Informationen erhalten. Peinlich ist es natürlich für Firmen, die für ein bestimmtes Produkt per Mail werben, und dann selbst das Konkurrenzprodukt einsetzen (z.B. Werbung für MDollar, geschrieben in Mutt, einem Linux-Mailclient)...

Wer sich noch genauer mit dem Format von Mails beschäftigen möchte, sollte sich den RFC 2076[15] ansehen (der erste RFC zum Thema war RFC 822[16], der 1982 eingereicht wurde).

4 SMTP

Der Mail-Header ist jetzt erklärt, doch eine Frage wurde bisher nur verdrängt: Wie kommen die Mails eigentlich beim Empfänger an?

Diesen Part übernimmt das Send Mail Transport Protocol, SMTP, eines der ersten Protokolle, die es im Internet[17], gab. Der erste RFC, der SMTP definierte, RFC 821[18], wurde bereits 1982 eingereicht.

Um einen Nutzer im Internet eindeutig zu adressieren, entwickelte man ein neues Adressierungsmuster: Man benutzt den Rechner, auf dem der Nutzer meistens anzutreffen ist (also eingeloggt ist), und seinen Login-Namen auf diesem als E-Mail-Adresse. Als Trenner fungiert das »@«-Zeichen.

Für´s Verschicken und Empfangen sind zwei unterschiedliche Protokolle zuständig, beide in der Anwendungsschicht des OSI-Schichtenmodells angesiedelt: SMTP zum Versenden und POP3 (auf Seite 31)) zum Empfangen von Mails. Dieses Kapitel erklärt nur SMTP, nicht POP3.

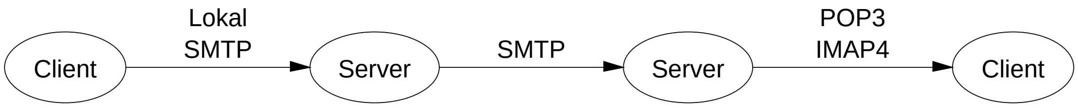
4.1 Grundlagen

E-Mails wandern bei SMTP i.A. über viele Server ehe sie ihr Ziel erreichen. Dies kommt noch aus der Zeit der vielen Mailboxnetze,

---

[15] `http://www.faqs.org/rfcs/rfc2076.html`
[16] `http://www.faqs.org/rfcs/rfc822.html`
[17] bzw. damals noch ARPAnet
[18] `http://www.faqs.org/rfcs/rfc821.html`

wo viele Internetverbindungen nicht permanent aktiv waren und so häufig andere Routen für die Mail-Übertragung genommen werden mussten.



SMTP ist ein sehr einfaches, ASCII-basiertes Protokoll, das heißt, auch Menschen können zum Beispiel mit dem »telnet«-Befehl das Protokoll nutzen, obwohl in der Praxis Mail-Server diese Aufgabe übernehmen.

4.2 Envelope-Header

Damit jeder Server weiß, wohin er eine eingegangene Mail weiterleiten soll, muss dieser in den sogenannten **Envelope-Header** schauen.

Dieser Header hat noch nichts mit der Mail selbst zu tun! Der Server interessiert sich nicht für den **Mail**-Header wie aus dem vorherigen Kapitel, nur der **Envelope**-Header ist entscheidend für das Ziel der Mail.

Dies ist vergleichbar mit „traditioneller" Post: Der Briefträger schaut nur auf den Umschlag, nicht aber in den Brief[19]. Somit kann in dem Brief ein ganz anderer Empfänger angegeben sein, ohne das der Postbote davon erfährt:

[19] Wenn man von korrupten Postboten einmal absieht. . . ;-)

4.3 Typischer Ablauf

Nun zu etwas Praxis: Wir werden eine Mail, die ein »foo@A.org« geschrieben hat, an »bar@B.org« weiterleiten. Dabei spielen wir den Mail-Server von »A.org«.

Also verbinden wir uns zuerst mit dem Standard-Port von SMTP, TCP-Port »25«, von Rechner »B.org«:

```
iblech@thestars theguide $ telnet B.org 25
Trying 200.233.42.11...
Connected to B.org.
Escape character is '^]'.
220 B.org Mailserver ready.
```

Dann authentifizieren wir uns bei »B.org« mit dem Kommando »HELO«[20]:

```
HELO A.org
250 B Hello A.org [199.232.41.10], pleased to meet you
```

Die »250« am Anfang der Antwort stellt dabei den Statuscode dar. »250«C bedeutet dabei „alles in Ordnung“, wie man an der englischen Antwort daneben auch erkennen kann.

Als nächstes erwartet »B.org« den Absender der E-Mail, und zwar den Envelope-Sender.

```
MAIL FROM: <foo@A.org>
250 2.1.0 <foo@A.org>... Sender ok
```

Damit »B.org« weiß, an wen er die Mail zustellen soll, kommt jetzt das Kommando RCPT (Abkürzung für **R**e**c**i**p**ien**t**):

```
RCPT TO: <bar@B.org>
250 2.1.5 <bar@B.org>... Recipient ok
```

Dieser Befehl kann auch mehrmals angewendet werden, um eine identische Mail an viele Empfänger zu schicken, ohne dass man jedesmal den Inhalt der Mail neu übertragen muss. Dies nutzen leider auch Spammer aus...

Nun erst wird mit der Übertragung der eigentlichen Mail begonnen:

[20] Tatsächlich wird man meistens »EHLO« verwenden. »EHLO« ist das »HELO« von ESMTP, Extended SMTP.

```
DATA
354 Enter mail, end with "." on a line by itself
From: Mister Foo <foo@A.org>
To: Bar Com <bar@B.org>
Subject: foobar
(Leerzeile)
blabla...
. (Punkt Enter)
250 2.0.0 h6EIf8O08948 Message accepted for delivery
```

Ab dem eingegebenen »DATA« kommt die eigentliche Mail, im Format wie im vorhergehenden Kapitel erläutert. Diese **Mail**-Header können hier **beliebig** gesetzt werden, ohne dass das irgendeine Software überprüft. Und ja, sie müssen **nicht** zwingend mit den Envelope-Headern übereinstimmen.

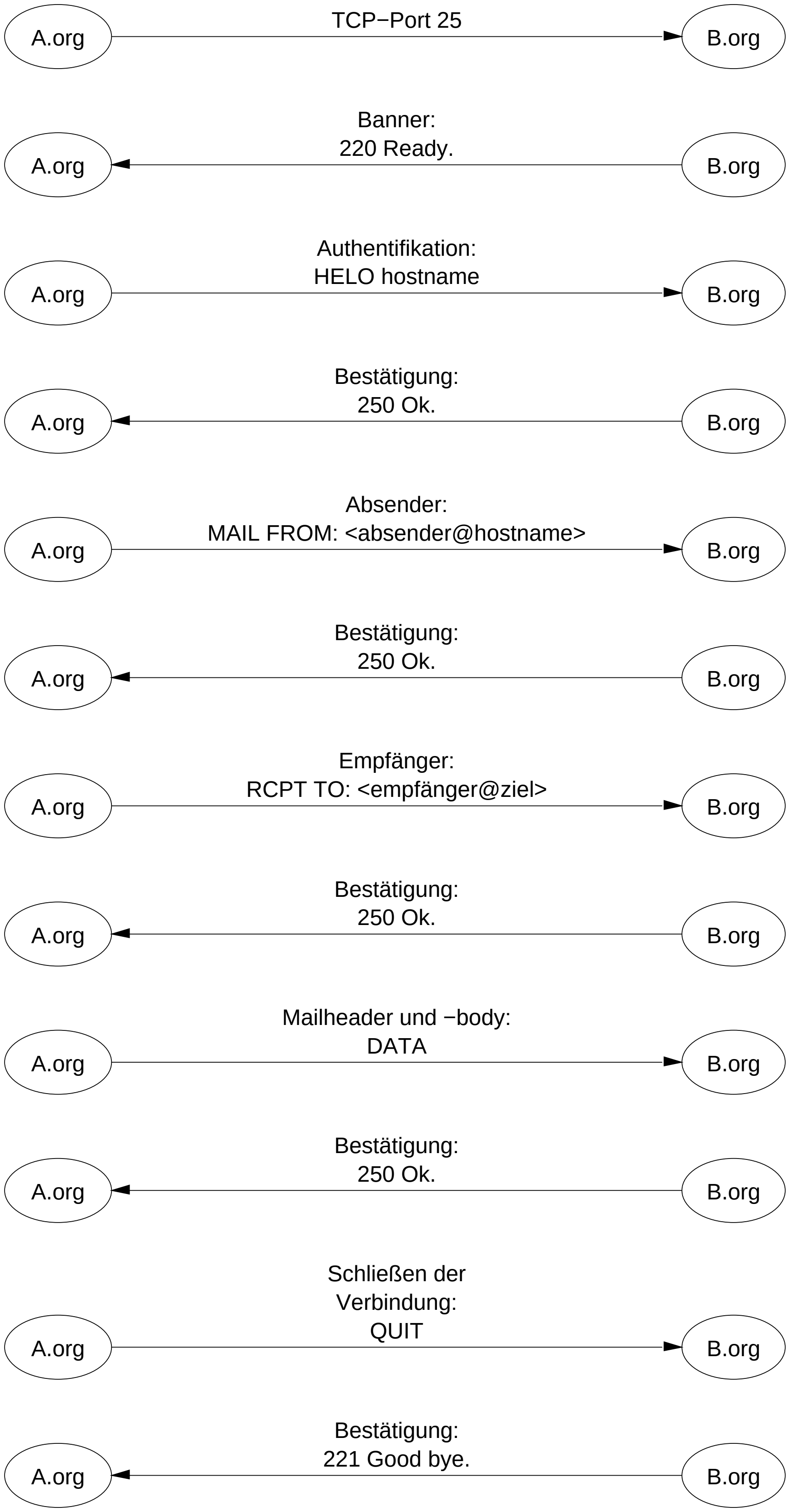
Theoretisch kann der Mail-Header sogar fehlen; Die Mail kommt trotzdem an, dafür sorgt ja der Envelope-Header.

Das Ende der Mail markiert ein einzelner Punkt auf einer Zeile[21]. Zum Schluss muss die Sitzung dann nur noch geschlossen werden:

```
QUIT
221 2.0.0 B.org closing connection
```

In einem kleinem Schema zusammengefasst sieht eine typische SMTP-Sitzung also so aus:

[21] Wie werden dann Mails verschickt, die einen einzelnen Punkt auf einer Zeile beinhalten? Der Server setzt einfach vor jede Zeile, die mit einem Punkt beginnt, einen weiteren Punkt, aus »..abc« wird also »...abc«. Programme, die die Mail dann später per POP3 oder IMAP4 abholen, löschen einfach jeden Punkt, der in der ersten Spalte einer Zeile steht. »s/ˆ“.//«


A.org
TCP−Port 25
B.org
Banner:
220 Ready.
Authentifikation:
HELO hostname
Bestätigung:
250 Ok.
Absender:
MAIL FROM: <absender@hostname>
Bestätigung:
250 Ok.
Empfänger:
RCPT TO: <empfänger@ziel>
Bestätigung:
250 Ok.
Mailheader und −body:
DATA
Bestätigung:
250 Ok.
Schließen der
Verbindung:
QUIT
Bestätigung:
221 Good bye.

4.4 Beispiel

Nun wollen wir die Theorie in die Praxis umsetzen:

Wir wollen eine E-Mail an »iblech-ml@gmx.de« im Namen von »billg@gates.gat«[22] schicken.

Es gibt zwei Möglichkeiten, zu welchem Mailserver wir connecten: Entweder, wir verbinden uns mit einem sogenannten **Open Relay**. Ein Open-Relay ist ein SMTP-Server, der jede Mail annimmt und dann entsprechend weiterleitet. Solche Open-Relays waren früher sehr beliebt und legitim, da ein Internetzugang noch sehr viel gekostet hat und so jemand anderes die Mails weiter verschickt hat (zum Beispiel ein Server einer Universität). Es kümmert sich also das Open Relay um den weiteren Transport der Nachricht. Da heutzutage aber auch Spammer Open Relays für ihre Zwecke missbrauchen, sind viele Relays in Blacklists eingetragen weshalb diese Möglichkeit wegfällt.

Aber wir können uns auch gleich mit dem SMTP-Server des Empfängers verbinden, diese Möglichkeit funktioniert immer. Nichts anderes machen nämlich die Mailserver der ISPs. Wie man diesen Server herausfindet, ist im Kapitel über DNS, im Abschnitt über MX-Records (auf Seite 111) beschrieben. Für jetzt soll es einfach genügen, wenn ich sage, dass der SMTP-Server von GMX »mx0.gmx.de« ist.

Sobald die Verbindung aufgebaut ist, werden wir uns als »localhost« identifizieren.

Bei der Eingabe des Envelope-Senders ist nun zu beachten, dass viele Mailserver die **Syntax** des Absenders überprüfen. Als Envelope-Sender ginge also zum Beispiel »hallo@-@dd« nicht durch. Deswegen müssen wir eine gültig **aussehende** E-Mail-Adresse als Absender wählen. Welcher Absender im **Mail**-Header dann steht, ist, wie schon erwähnt, vollkommen egal.

Also geben wir ein:

```
iblech@thestars theguide $ telnet mx0.gmx.de 25
Trying 213.165.64.100...
Connected to mx0.gmx.de.
```

[22] Die Domain »gates.gat« gibt es (zumindest bei der ICANN) nicht; da diese aber nur im **Mail**-Header (und nicht im Envelope-Header, der vielleicht auf syntaktische Korrektheit hin überprüft wird) zu finden sein wird, ist dies egal.

```
Escape character is '^]'.
220 {mx039} GMX Mailservices ESMTP
EHLO localhost
250-{mx039} GMX Mailservices
250-8BITMIME
250 ENHANCEDSTATUSCODES
MAIL FROM: <koennte_existieren@realer_host.com>
250 2.1.0 {mx039} ok
RCPT TO: <iblech-ml@gmx.de>
250 2.1.5 {mx039} ok
DATA
354 {mx039} Go ahead
From: The Imperor <billg@gates.gat> $$$
To: You! Yes, you! <html>
Subject: Hmm...
(Leerzeile)
muhahahaha...
.
250 2.6.0 {mx039} Message accepted
QUIT
221 2.0.0 {mx039} GMX Mailservices
Connection closed by foreign host.
iblech@thestars theguide $
```

Wenn ich nun in mein Postfach schaue, werde ich eine Mail von »The Imperor <billg@gates.gat¿ $$$« an »You! Yes, you! <html¿« in meinem Postfach finden.

Wenn sich Mails also **so** einfach fälschen lassen – wie stellt man dann trotzdem eine sichere Kommunikation her? Man muss jede ausgehende Mail digital verschlüsseln und signieren. In Mailclients integrierbare Programme wie GnuPG[23] übernehmen diese Aufgabe. Eine sehr gute Einführung zu GnuPG gibt ein Bericht von Pro-Linux[24].

[23] http://www.gnupg.org/
[24] http://www.pl-berichte.de/berichte/gnupg.html

4.5 Befehlsübersicht

Befehl	Wirkung
»HELO hostname«	Authentifizierung (SMTP)
»EHLO hostname«	Authentifizierung (ESMTP)
»MAIL FROM: <sender@host¿«	Envelope-Sender
»RCPT TO: <empfaen-ger@host¿«	Envelope-Empfänger
»DATA«	Beginn der Mail
».«	Ende der Mail
»QUIT«	Terminierung der Verbindung

5 POP3

Das Personal Office Protocol, POP3, definiert in RFC 1939[25], ist praktisch das Gegenstück zu SMTP. Auch auf ASCII basierend, holt es die Mails vom Server ab anstatt sie zu versenden.

Auf Extended POP3, EPOP3, wird hier nicht eingegangen, da es im Wesentlichen nur neue Authentifizierungsmöglichkeiten unterstützt. Es gibt keine fundamentalen Unterschiede zwischen POP3 und EPOP3.

5.1 Grundlagen

Bei POP3 werden die Mails, welche zuvor via SMTP versendet wurden, auf dem Server so lange gespeichert, bis sie vom Client gelöscht werden[26].

Ohne POP3 (oder einem ähnlichen Protokoll) müsste der Client permanent online sein, um E-Mails zu erhalten, was besonders früher wegen der hohen Onlinekosten wirklich nicht erwünscht war. . .

5.2 Typische POP3-Sitzung

POP3 ist eines der einfachsten Protokolle. Im Gegensatz zu SMTP, wo es Hunderte von Statuscodes gibt, findet man bei POP3 nur

[25] `http://www.faqs.org/rfcs/rfc1939.html`
[26] ...oder das Quota erreicht ist. . .

»+OK« und »-ERR« vor. Auch gibt es bei POP3 im Gegensatz zu zum Beispiel IMAP wenig Interaktivität; Der Client verlangt etwas (Liste der auf dem Server gespeicherten Mails oder eine gewünschte Mail), und der Server antwortet sofort.

Als Beispiel dient das einfache Abrufen samt anschließendem Löschen der Mails **ohne** einem Mail User Agent (MUA, z.B. »mutt«, »mailx« oder »kmail«).

Zuerst verbinden wir uns deshalb mit dem POP3-Server, der unsere Mails speichert, auf TCP-Port »110«. Da wir meine Mails abholen wollen, ist der entsprechende POP3-Server »pop3.web.de«:

```
iblech@thestars theguide $ telnet pop3.web.de 110
Trying 217.72.192.134...
Connected to pop3.web.de.
Escape character is '^]'.
+OK WEB.DE POP3-Server
```

»WEB.DE POP3-Server« ist dabei der sogenannte Banner. In der Banner-Zeile wird oft auch die eingesetze Version des POP3-Servers übertragen.

Dann loggen wir uns ein:

```
USER iblech
+OK Bitte Kennwort eingeben/enter password
PASS Th1s_1s_4_VeRy_S3cRe7_p4s5w0rd
+OK Postfach bereit/mailbox locked and ready
```

Nun können wir eine Liste der auf dem Server gespeicherten Messages abrufen:

```
LIST
+OK
1 1882
2 5665
.
```

Um jetzt z.B. die zweite Mail herunterzuladen, geben wir ein:

```
RETR 2
+OK Nachricht folgt/message follows
From: Some Body <some@body.com>
To: Ingo Blechschmidt <iblech@web.de>
Subject: Buy now

Hi man,

...
.
```

Man erhält also den schon von SMTP bekannten Mail-Header samt Body (wieder mit einer Leerzeile getrennt), als Terminierung wird ein einzelner Punkt auf einer Zeile (».«) benutzt.

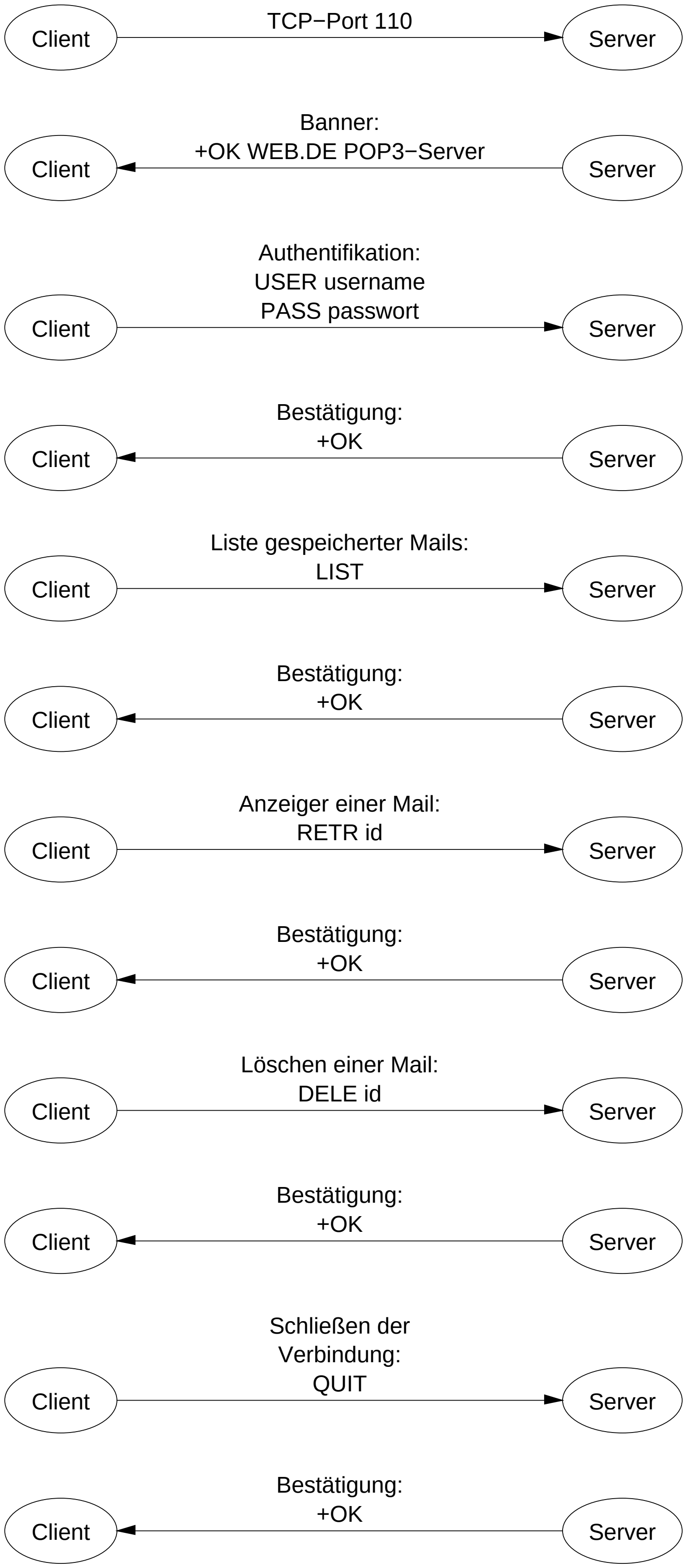
Jetzt, wo der Client, also wir, die Mail erhalten hat, können wir die Mail löschen:

```
DELE 2
+OK Nachricht wurde geloescht/message deleted
```

Um dann die Verbindung zu schließen tippen wir:

```
QUIT
+OK
iblech@thestars theguide $
```

5.3 Protokollablauf



```
iblech@thestars theguide $ telnet pop3.web.de 110
Trying 217.72.192.134...
Connected to pop3.web.de.
Escape character is '^]'.
+OK WEB.DE POP3-Server
USER iblech
+OK Bitte Kennwort eingeben/enter password
PASS Ultr4 1337 p455w0rd
+OK Postfach bereit/mailbox locked and ready
LIST
+OK
1 1882
2 5665
3 2430
4 1790
5 5079
.
RETR 1
+OK Nachricht folgt/message follows
Received: from [204.126.2.42] (helo=gentoo.org)
    by mx16.web.de with smtp (WEB.DE 4.99 #401)
    id 19crxk-0006K8-00
    for iblech@web.de; Wed, 16 Jul 2003 21:29:16 +0200
Received: (qmail 8463 invoked by uid 1002); 16 Jul 2003
19:28:37 -0000
Mailing-List: contact gentoo-user-help@gentoo.org; run
by ezmlm
Precedence: bulk
List-Id: Gentoo Linux mail <gentoo-user.gentoo.org>
Reply-To: gentoo-user@gentoo.org
X-BeenThere: gentoo-user@gentoo.org
...
.
DELE 1
+OK Nachricht wurde geloescht/message deleted
QUIT
+OK
Connection closed by foreign host.
iblech@thestars theguide $
```

Das war´s auch schon. So kann man auch ohne installierten MUA seine Mails abrufen und löschen, zum Beispiel wenn eine 100 MeB große Mail auf dem Server ist und man gerade keine Lust hat, wegen ISDN 243 Minuten zu warten.

5.4 Befehlsübersicht

Befehl	Wirkung
»USER benutzername«	Login mit Benutzername...
»PASS P455w0r7«	...und Passwort
»LIST«	Liste aller auf dem Server gespeicherten Mails
»STAT«	Anzahl der gespeicherten Mails samt ihrer Größe in Bytes
»RETR id«	Holt eine Mail
»DELE id«	Löscht eine Mail
»QUIT«	Beendet die Verbindung

6 IMAP4

Wie POP3 dient das Internet Message Access Protocol, IMAP, definiert in RFC 1730[27] dazu, die auf dem Server gespeicherten Mails abzurufen. Allerdings bleiben bei IMAP die Mails auch üblicherweise auf dem Server, das heißt der Server übernimmt die **permanente** Speicherung der Mails.

Dies ist besonders für Nutzer, die oft an verschiedenen Computern arbeiten, nützlich, da sie nun auf alle ihrer Mails von jedem Computer aus Zugriff haben.

Auch gibt es bei IMAP (server-seitige) Ordner, ein Fremdwort für POP3.

6.1 Technische Implementierung

Direkt nach dem Verbinden auf den Standardport »143« (TCP) des IMAP4-Servers (im Beispiel »imap.web.de«) empfängt man eine Bannermeldung:

```
iblech@thestars theguide $ telnet imap.web.de 143
Trying 217.72.192.135...
Connected to imap.web.de.
Escape character is '^]'.
* OK WEB.DE IMAP4-Server
```

[27] http://www.faqs.org/rfcs/rfc1730.html

Als nächstes loggen wir uns ein:

```
A001 LOGIN "iblech" "pa5sword"
A001 OK User logged in
```

Da bei IMAP theoretisch mehrere Befehle gleichzeitig ausgeführt werden können, bedient man sich von IDs, die jedem Befehl voranstehen müssen (hier »A001«). Die ID kann mehr oder weniger frei gewählt werden, es dürfen nur keine Leerzeichen oder andere Sonderzeichen in der ID enthalten sein. Auch muss die nächste ID **nicht** unbedingt „höher“ (zum Beispiel »A002«) sein.

Nun können wir einen Ordner auswählen:

```
A002 SELECT "inbox"
* 1 EXISTS
* OK [UNSEEN 1] Message 1 is first unseen
* OK [PERMANENTFLAGS (\Deleted \Seen \Answered)]
* OK [UIDVALIDITY 1]
* FLAGS (\Answered \Flagged \Deleted \Seen \Draft)
A002 OK Completed [READ-WRITE]
```

»inbox« ist der Standard-Ordner, „Posteingang“. Aus der Antwort können wir sehen, dass eine Nachricht noch neu ist (»UNSEEN 1«).

Bevor wir allerdings die Nachricht herunterladen, können wir schauen, wie groß die Mail (wie bei POP3 angegeben in Octets, also Bytes) ist:

```
A003 FETCH 1 RFC822.SIZE
* 1 FETCH (RFC822.SIZE 2247)
A003 OK Completed
```

Nun können wir die Übertragung des Headers der Mail einleiten:

```
A004 FETCH 1 RFC822.HEADER
* 1 FETCH (RFC822.HEADER {1446}
Received: from [204.126.2.42] (helo=gentoo.org)
        by mx22.web.de with smtp (WEB.DE 4.99 #420)
        id 19iCSL-0008FU-00
        for iblech@web.de; Thu, 31 Jul 2003 14:22:53 +0200
Received: (qmail 1148 invoked by uid 1002); 31 Jul 2003
Mailing-List: contact gentoo-user-help@gentoo.org
```

```
From: "Fellipe Weno" <fellipe@eiconbrasil.com.br>
To: <gentoo-user@gentoo.org>
Subject: Re: [gentoo-user] winex dont run

)
A004 OK Completed
```

Hier wird das Ende, **nicht** wie bei SMTP, POP3 oder NNTP mit einem Punkt, sondernmit »)« (runde Klammer zu) markiert.

Auch der Body der Mail ist schnell geholt:

```
A005 FETCH 1 BODY.PEEK[TEXT]
* 1 FETCH (BODY[TEXT] {873}
i dont need notepad i wanna run diablo 2 game... but i get
it work i get source from cvs tree and compile it and its
work fine =)
thank's

--
gentoo-user@gentoo.org mailing list

)
A005 OK Completed
```

Hätten wir die gesamte Mail auf einmal holen wollen, hätten wir auch

```
A005 FETCH 1 RFC822
```

benutzen können.

Nun kann die Mail gelöscht (obwohl das bei IMAP, wie oben erwähnt, eher unüblich ist) werden:

```
A006 STORE 1 +FLAGS (\Deleted)
* 1 FETCH (FLAGS (\Deleted))
A006 OK Completed
```

Mit diesem Befehl setzen wir den Status der Mail auf „zum Löschen bereit“. Mit

```
A007 EXPUNGE
* 1 EXPUNGE
OK EXPUNGE completed
```

wird die Mail dann endgültig gelöscht. Nachdem die Mail gelöscht ist, verschieben sich auch die Nummern der Mails nach vorne, das heißt Nachricht Nummer 2 ist nach der Löschaktion die erste Nachricht.

Mit »LOGOUT« beenden wir die Verbindung:

```
A008 LOGOUT
* BYE LOGOUT received
A008 OK Completed
Connection closed by foreign host.
iblech@thestars theguide $
```

6.2 Befehlsübersicht

Befehl	Wirkung
»LOGIN "name" "pass"«	Login
»SELECT "ordner"«	In Ordner wechseln
»FETCH nummer RFC822.SIZE«	Mailgröße anzeigen
»FETCH nummer RFC822.HEADER«	Mailheader anzeigen
»FETCH nummer BODY.PEEK[TEXT]«	Mailbody anzeigen
»FETCH nummer RFC822«	Gesamte Mail anzeigen
»STORE nummer +FLAGS ("Deleted)«	Mail zum Löschen markieren
»EXPUNGE«	Alle als gelöscht markierten Mails löschen
»LOGOUT«	Verbindung schließen

6.3 Abschluss

Mit IMAP steht Telnettern nun ein weiteres Protokoll zum Abholen von Mails zur Verfügung. Zur Übersicht nochmal eine komplette Beispielsitzung:

```
iblech@thestars theguide $ telnet imap.web.de 143
Trying 217.72.192.135...
Connected to imap.web.de.
Escape character is '^]'.
```

```
* OK WEB.DE IMAP4-Server
A LOGIN "iblech" "r1cht1g l4ng3s P4ssw0rt"
A OK User logged in
B SELECT "inbox"
* 14 EXISTS
* OK [UNSEEN 13] Message 1 is first unseen
* OK [PERMANENTFLAGS (\Deleted \Seen \Answered)]
* OK [UIDVALIDITY 1]
* FLAGS (\Answered \Flagged \Deleted \Seen \Draft)
B OK Completed [READ-WRITE]
C FETCH 3 RFC822
* 3 FETCH (FLAGS (\Seen) RFC822 {4147}
Received: from [146.82.138.6] (helo=murphy.debian.org)
        by mx06.web.de with esmtp (WEB.DE 4.99 #420)
        id 19iCYx-0007Kd-00
        for iblech@web.de; Thu, 31 Jul 2003 14:29:43 +0200
(...)
To: debian-user-german@lists.debian.org
Subject: Re: history von: "dpkg -i"
Date: Thu, 31 Jul 2003 14:21:30 +0200
Message-ID: <878yqezy2d.fsf@alhambra.bioz.unibas.ch>
User-Agent: Gnus/5.1002 (Gnus v5.10.2) Emacs/21.3 (gnu/linux)

Marko Schulz <4mschulz@informatik.uni-hamburg.de> schrieb:
(...)

dpkg-reconfigure apt-listchanges

Gru=DF, Frank
)
C OK Completed
D LOGOUT
* BYE LOGOUT received
D OK Completed
Connection closed by foreign host.
iblech@thestars theguide $
```

7 NNTP

Das Network News Transport Protocol, NNTP, definiert in RFC 977[28], ist das für die Verbreitung von Usenet-Messages (News) zuständige Protokoll.

[28] http://www.faqs.org/rfcs/rfc977.html

Auch wenn heute unter Lamer-Kreisen die Web-basierten Foren das Usenet immer mehr verdrängen, wird es doch noch von sehr vielen Leuten besucht und ist der erste Ort wo man fragen sollte falls zum Beispiel Computer-Probleme[29] auftauchen.

Um die Beispiele dieses Kapitels nachzuvollziehen kann man sich bei individual.net[30], einem Dienst der Zentraleinrichtung für Datenverarbeitung (ZEDAT), kostenlos einen Account holen, auch wenn die Registrierung von Menschenhand geprüft wird und es deswegen ein bis zwei Tage dauern kann, bis man seine Login-Daten erhält.

7.1 Design

NNTP ist ein meiner Meinung nach sehr geniales Protokoll, da es völlig dezentral orientiert ist:

Eine Usenet-Mail, hochgeladen auf irgendeinen Server des NNTP-Netzwerkes, wird innerhalb von Stunden an die anderen Server weitergeleitet, sodass sie sehr schnell netzweit abrufbar ist (siehe dazu die Beschreibung des »IHAVE«-Kommandos (auf Seite 48)).

Bei NNTP besteht eine Message genau wie bei „normalen“ RFC822-Mails (auf Seite 20) aus einem Header und einem Body.

7.2 Beispielsitzung

Ähnlich wie die bisher besprochenen Protokolle der Anwendungsschicht basiert NNTP auf ASCII, das heißt man kann ohne Probleme via Telnet Posten (Usenet-Messages versenden) oder online lesen.

Zunächst verbinden wir uns zuerst mit dem News-Server, im Beispiel »news.individual.net« auf dem Standard-NNTP-Port »119« (TCP).

```
iblech@thestars theguide $ telnet news.individual.net 119
Trying 130.133.1.4...
Connected to news.individual.net.
Escape character is '^]'.
```

[29] In aller Regel MDollar-Probleme

[30] http://news.individual.de/

```
200 The server welcomes \
  251.8-dial.augustakom.net (80.81.8.251). \
  Authorization required for reading and posting.
```

Wir folgen dem Aufruf des Banners und loggen uns ein:

```
AUTHINFO SIMPLE blechschmidt p4ssw0rt
281 Authentication accepted. (UID=105138)
```

Jetzt könnten wir mit »LIST NEWSGROUPS« alle auf diesem Server verfügbaren Newsgruppen anzeigen lassen. Da dies wegen der Vielzahl von Gruppen lange dauern kann, zeigen wir gleich die IDs der letzten Artikel der Newsgruppe »de.comp.os.unix.shell« an:

```
LISTGROUP de.comp.os.unix.shell
211 Article list follows
17226
17228
17229
17230
(...)
.
```

Wie bei SMTP und POP3 wird das Ende der Übertragung mit einem einzelnen Punkt auf einer Zeile für sich markiert.

Um nun die Message mit der ID »17226« abzuzeigen, benutzen wir das »ARTICLE«-Kommando:

```
ARTICLE 17226
220 17226 <pan.2004.03.18.05.41.06.452379@tirolinux.net>
article
Path: uni-berlin.de!fu-berlin.de!newsfeed01.chello.at!...
From: Daniel Knabl <news@tirolinux.net>
Subject: Re: problem mit if then else in script
Message-ID: <pan.2004.03.18.05.41.06.452379@tirolinux.net>
Newsgroups: de.comp.os.unix.shell
Xref: uni-berlin.de de.comp.os.unix.shell:17226

Am Wed, 17 Mar 2004 15:59:47 +0100 schrieb Stefan Lagotzki:

> Naja, 62 Seiten sind es hier. Wenn Du ein gescheites
> Linux/Unix hast, sollte ein LaTeX mit dabei sein,
> dann mach mal folgendes:
>
> man bash -Tdvi > bash.dvi; dvipdfm bash.dvi
(...)
.
```

ein. Auch hier wird das Ende der Nachricht wieder auf die gleiche Weise markiert. Hätten wir nur den Header der Message herunterladen wollen, hätten wir auch das »HEAD«-Kommando benutzen können:

```
HEAD 17226
```

eingeben können. Wären wir nur am Inhalt interessiert, würden wir

```
BODY 17226
```

benutzen.

Um einen Artikel zu posten, wechseln wir zuerst in die Newsgroup »de.test« mit dem Kommando »GROUP« (alternativ könnten wir auch »LISTGROUP« benutzen, um eine Gruppe auszuwählen) und benutzen dann den »POST«-Befehl:

```
GROUP de.test
211 16440 460131 477610 de.test
POST
340 Ok, recommended ID <2q8f38FseklaU1@uni-berlin.de>
From: Tester <test@test.com>
Subject: Test 42 Pi
Newsgroups: de.test

DaTest
.
240 Article posted <2q8f38FseklaU1@uni-berlin.de>
```

Hier können wir auch (wie im Beispiel gemacht) den Absender faken. Da allerdings bei den meisten NNTP-Servern die zum Posten verwendete User-ID in den Header aufgenommen wird, ist das Zurückverfolgen (anders als bei SMTP) sehr einfach, man muss nur in den Header der Nachricht schauen.

Zur Kontrolle schauen wir uns nun die Message an. Dazu verwenden wir wieder den »ARTICLE«-Befehl, diesmal allerdings statt der Artikel-Nummer die Message-ID als Parameter:

```
ARTICLE <2q8f38FseklaU1@uni-berlin.de>
220 0 article <2q8f38FseklaU1@uni-berlin.de>
Path: uni-berlin.de!not-for-mail
From: Tester <test@test.com>
Newsgroups: de.test
Subject: Test 42 Pi
Date: 8 Sep 2004 13:17:54 GMT
Lines: 1
Message-ID: <2q8f38FseklaU1@uni-berlin.de>
X-Trace: news.uni-berlin.de
DfHQkh6rt6WDbJ7PzCw23wmZzfL4gxSj2x1I8SWKdPoA0xh2AK
Xref: uni-berlin.de de.test:477612

DaTest
.
```

Von wesentlicher Bedeutung ist der (in diesem Beispiel) vom Server automatisch hinzugefügte Header »Message-ID«. Diese ID identifiziert jede Nachricht (sowohl Usenet-Mails als auch „normale“ Mails) **eindeutig**. Die Message-ID setzt sich üblicherweise aus zufälligen Zeichenketten, der aktuellen Zeit und dem User-Namen des Posters zusammen. Ohne die Eindeutigkeit der Message-ID ist eine Verbreitung der Nachricht über mehrere Server nicht denkbar.

Die Verbindung schließen wir mit »QUIT«:

```
QUIT
205 .
iblech@thestars theguide $
```

7.2.1 Zusammenfassung

Hier noch mal die einzelnen Befehle, diesmal ohne die Antworten des Servers:

```
AUTHINFO SIMPLE benutzername password
LISTGROUP gruppenname
ARTICLE nummer
GROUP gruppenname
POST
(Usenet-)Mail-Header und -Body
QUIT
```

Und, zum Überblick, eine vollständige Beispielsitzung:

```
iblech@thestars theguide $ telnet news.individual.net nntp
Trying 130.133.1.4...
Connected to news.individual.net.
Escape character is '^]'.
200 The server welcomes \
  251.8-dial.augustakom.net (80.81.8.251). \
  Authorization required for reading and posting.
AUTHINFO SIMPLE blechschmidt passwort
281 Authentication accepted. (UID=105138)
LIST
215 Newsgroups in form "group high low flags".
alt.2600 0000207579 0000180928 y
alt.2600.hackerz 0000069040 0000065852 y
.
LISTGROUP de.test
211 Article list follows
428171
428172
.
POST
340 Ok, recommended ID <2q8g23FrhmpvU1@uni-berlin.de>
From: Tester <test@test.com>
Subject: Test 1337
Newsgroups: de.test

Gentoo GNU/Linux! :)
.
240 Article posted <2q8g23FrhmpvU1@uni-berlin.de>
ARTICLE <2q8g23FrhmpvU1@uni-berlin.de>
220 0 article <2q8g23FrhmpvU1@uni-berlin.de>
Path: uni-berlin.de!not-for-mail
From: Tester <test@test.com>
Newsgroups: de.test
Subject: Test 1337
Date: 8 Sep 2004 13:34:28 GMT
Lines: 1
Message-ID: <2q8g23FrhmpvU1@uni-berlin.de>
X-Trace: news.uni-berlin.de
qjy7sH6+HGAVA4eZVYxPgww5srYnQAQiisjBZBnq0cNAOup/R+
Xref: uni-berlin.de de.test:477614

Gentoo GNU/Linux! :)
.
QUIT
205 .
Connection closed by foreign host.
iblech@thestars theguide $
```

7.3 Besondere Header

NNTP geht jedoch über eine bloße Ansammlung von „normalen“ Mails hinaus: Es gibt einige Header-Felder, die die Verteilung der Usenet-Mails regeln.

7.3.1 »Newsgroups«

Über den »Newsgroups«-Header wird festgelegt, ich welchen Gruppen die Nachricht erscheint. Getrennt werden mehrere Gruppen (zum Beispiel beim Cross-Posting (unter Cross-Posting versteht man das verschicken einer Nachricht an mehrere (meistens thematisch verwandte) Gruppen) durch »,« (Komma):

```
Newsgroups: de.comp.text.tex, de.comp.text.genuineword
```

7.3.2 »Followup-To«

Der »Followup-To«-Header entscheidet, in welche Newsgroup Antworten gehen. Dies ist insbesondere beim Cross-Posting nützlich (und wird im Allgemeinen auch vom Poster erwartet), da nur so alle Antworten gesammelt in eine Newsgroup gehen.

Besonders ist das Ziel »poster«, in dem Fall gehen die Antworten direkt an den Sender der Nachricht, per Mail:

```
From: Ingo Blechschmidt <iblech@web.de>
Subject: Suche XYZ
Followup-To: poster
```

7.3.3 »Path«

Der »PATH«-Header zeigt die Weg der Nachricht über verschiedene Server des News-Netzes auf. Dieser Header wird in der Regel automatisch vom News-Server zur Mail hinzugefügt. Getrennt werden einzelne Stationen mit »!« (Ausrufezeichen).

```
Path: uni-berlin.de!fu-berlin.de!\
  newsfeed01.chello.at!news.chello.at.POSTED!\
  53ab2750!not-for-mail
```

7.3.4 »Approved«

Bei moderierten Gruppen (Gruppen, deren jede Nachricht erst von einem Moderator bestätigt werden muss, damit zum Beispiel keine Flamewars ausbrechen oder Off-Topic-Posts die Regel werden) ist der »Approved«-Header nötig, um die Moderation zu **umgehen**. Nachrichten mit gültigem »Approved«-Header werden als schon moderiert angesehen und umgehen damit den Moderationsmechanismus. Mehr dazu siehe Google.

7.3.5 »Control«

Besondere Usenet-Mails sind „Control-Messages". Mit ihnen können zum Beispiel vorhandene Nachricht gelöscht („gecancelt") werden, neue Newsgruppen angelegt und entfernt werden, etc.

Jede Control-Nachricht muss den »Control«-Header besitzen. Um zum Beispiel die (eigene) Nachricht mit der Message-ID »<abc@host.org>« zu canceln, benutzt man:

```
Control: cancel <abc@host.org>
```

Dabei ist darauf zu achten, dass die „Mail" vom gleichen Absender verschickt wird, der »From«-Header muss also mit dem »From«-Header des Postings, das gelöscht werden soll, übereinstimmen[31].

Mit Control-Nachrichten kann man auch neue Newsgruppen erstellen, mehr hierzu kann man in RFC 1036[32] nachlesen.

7.4 Austausch zwischen den Servern

Bisher wurde noch nicht erklärt, wie eine Nachricht auf Server »A« an alle anderen Server weitergeleitet wird.

Server »A« könnte sich natürlich als den Original-Poster ausgeben, zu jedem anderen Server der Erde connecten und dort wieder via »POST« die Nachricht posten. Es kommt jedoch ein sehr viel effizienteres Verfahren zum Einsatz, »IHAVE«:

[31] Tatsächlich wird zuerst der »Sender«-Header überprüft, ist der nicht vorhanden wird zum »From« übergegangen.

[32] http://www.faqs.org/rfcs/rfc1036.html

Angenommen, jemand postet auf »A« eine Mail mit der Message-ID »<r4ndom@host>«. Dann wird »A« zu einem oder zwei anderen Servern connecten und dort »IHAVE« ausführen:

```
IHAVE <r4ndom@host>
335 send article to be transferred.
From: ...
(...)

Hallo, ...
(...)
.
235 article transferred ok
```

Hat der Server den Artikel schon, kann »A« diesen Umstand an einer verändeten Rückmeldung erkennen:

```
IHAVE <r4ndom@host>
435 article not wanted - do not send it
```

Dieses Spiel führt jeder Server in einem bestimmten Intervall aus, so dass nach kürzester Zeit die Nachricht auf allen (an das Netz angeschlossenen) Servern verfügbar ist.

7.5 Befehlsübersicht

Befehl	Wirkung
»AUTHINFO SIMPLE user pass«	Login
»LIST NEWSGROUPS«	Liste aller Newsgroups des Server anzeigen
»LISTGROUP gruppenname«	Artikel-IDs einer Gruppe anzeigen und in die angegebene Gruppe wechseln
»GROUP gruppenname«	In angegebene Gruppe wechseln
»ARTICLE nummer« oder »ARTICLE <msg-id>«	Gewählte Nachricht anzeigen
»HEAD nummer« oder »HEAD <msg-id>«	Header der gewünschten Nachricht holen
»BODY nummer« oder »BODY <msg-id>«	Body der angegebenen Nachricht holen
»POST«	Eine Nachricht senden
»IHAVE <msg-id>«	Eine Nachricht verteilen
»QUIT«	Sitzung beenden

7.6 Abschluss

Wie man sieht, ist NNTP trotz der sehr stark ausgeprägten Dezentralisierung noch sehr einfach zu bedienen, Lesen und Posten von Usenet-News sollte jetzt auch via Telnet kein Problem mehr sein. Zum Testen der neu erworbenen Kenntnise sollte man allerdings Newsgruppen wie »de.test« verwenden, um andere Leser nicht mit falschen Headern zu stören ;-). . .

Übrigens ist auch lynx (auf Seite 117) ein einfach zu bedienender Newsreader, was zwei Screenshots beweisen sollen:

```
 ____________________________________________________
|<<<          Newsgroup de.sci.mathematik (p1 of 4)|
|      de.sci.mathematik, Messages 142216-142245    |
|                                                   |
|   (Frühere Messsages...)                          |
|                                                   |
|   Articles in de.sci.mathematik                   |
```

```
|     * "Re: [HJ]Re: Zahl^0" - "Peter Niessen"      |
|     * "Re: [HJ]Re: Zahl^0" - "Peter Niessen"      |
|     * "[FAQ] <2002-04-09> de.sci.mathematik" -    |
|       Tjark Weber                                 |
|     * "Re: [HJ]Re: Zahl^0" - "Peter Niessen"      |
|     * "Re: Stammfunktion von                      |
|       \int{\sqrt{1+sin(x)^2} dx}" - "Hermann      |
|       Kremer"                                     |
|     * "Re: [HJ]Re: Zahl^0" - Gabriela             |
|       Salvisberg                                  |
|     * "Re: [HJ]Re: Zahl^0" - "Hermann Kremer"     |
|     * "Re: Spektralanalyse" - Ralf Beyer          |
|     * "Re: Polynomdivision" - Sebastian Kapfer    |
|     * "Re: Umordnung v. unendl. Reihen" -         |
|       "Christian Stapfer"                         |
|-mehr-                                            |
|   news://news.cis.dfn.de/de.sci.mathematik/14218 |
|-------------------------------------------|

 -------------------------------------------
|<<<  [FAQ] <2002-04-09> de.sci.mathematik (p1 of 3)|
|        [FAQ] <2002-04-09> de.sci.mathematik      |
|                                                  |
|   From: Tjark Weber <tjark.weber@gmx.de>         |
|   Reply to: Tjark Weber                          |
|   Date: Tue, 22 Jul 2003 22:10:02 +0000          |
|   Organization: Akallabeth FAQ-posting-service   |
|   Newsgroups:                                    |
|         de.sci.mathematik                        |
|   Followup to: newsgroup                         |
|Posted-By: auto-faq 3.3.thh (Perl 5.006)          |
|Last-modified: 2002-04-09                         |
|Posting-frequency: weekly                         |
|                                                  |
|Herzlich willkommen in der Newsgroup              |
|de.sci.mathematik!                                |
|                                                  |
|                                                  |
|Worum geht es hier?                               |
|===================                               |
|                                                  |
|-mehr- mailto:tjark.weber@gmx.de                  |
|-------------------------------------------|
```

8 HTTP

Das Hypertext Transport Protocol ist verantwortlich für das Web, was ja schon an vielen URLs (Uniform Ressource Locators), zum Beispiel »http://linide.sf.net/«, erkennbar ist.

Wie SMTP (auf Seite 24) ist es ein ASCII-basiertes Protokoll, das heißt man kann »telnet« benutzen, um im Internet zu surfen...

8.1 Versionen

Heute werden HTTP 1.0, definiert in RFC 1945[33], und 1.1, definiert in RFC 2616[34], benutzt.

HTTP/1.1 hat einige Verbesserungen gegenüber HTTP/1.0 erhalten, zum Beispiel Keep-Alive[35] und virtuelle Hosts. Beiden dieser Features ist ein eigener Abschnitt gewidmet.

Der Standard-TCP-Port für HTTP (1.0 und 1.1) ist »80«.

Außerdem gibt es noch HTTP-Proxies[36]. Dort wird, je nach Zielhost, sowohl HTTP/1.0 als auch HTTP/1.1 verwendet. Oft benutzte Ports sind »3128«, »8080« und »79«; Einen Standardport gibt es nicht.

8.2 Adressen

Um HTTP zu verstehen, muss man wissen, wie Internetseiten eigentlich addressiert werden.

Da gibt es zum einen die

- „Lamer-Schreibweise“: »www.foobar.de«. Schlecht. Hier wird keine Adresse beschrieben, sondern lediglich einen Hostnamen.

---

[33] `http://www.faqs.org/rfcs/rfc1945.html`

[34] `http://www.faqs.org/rfcs/rfc2616.html`

[35] Auch HTTP/1.0 besitzt schon Keep-Alive, allerdings ist der Umgang mit Keep-Alive im Zusammenspiel mit (HTTP/1.0) Proxies sehr schwierig.

[36] Proxies im Allgemeinen leiten en Traffic, den sie erhalten, mehr oder weniger gefiltert an das Ziel weiter. Die Antwort wird dann an den Client zurückgeschicht

- Dann gibt es die URLs: »http://linide.sf.net/index.xhtml«. Gut. Alle für den Browser wichtigen Bestandteile sind vorhanden: Protokoll (HTTP oder FTP), Host, und Pfad (»/index.html« im Beispiel).

- Und schließlich: URIs, Uniform Ressource Identifiers. Diese sind ähnlich wie die URLs, haben aber den Vorteil, dass als Protokolle nicht nur HTTP und FTP möglich sind. Ihre generelle Form ist »protokoll://host:portpfad«. Der Pfad ist wie bei URLs Unix-like, also **keine** MDollar-Backslashes (»“«)!

  Wenn man nun allerdings einem normalen HTTP-Server eine URL vorsetzt, kann er damit nichts anfangen: Schließlich soll er nicht als Proxy (auf Seite 55) fungieren, sonden nur seine eigenen Seiten ausliefern.

8.3 HTTP/1.0

Deswegen benutzt man bei HTTP nur den Pfad (samt dem ersten »/«).

Ein einfacher »GET«-Request („zeig mir den Inhalt (Quelltext) einer Seite an!“) für die URL »http://pro-linux.de/« sieht dann so aus:

- Zuerst verbinden wir uns mit dem Host »pro-linux« auf TCP-Port 80:

```
iblech@thestars theguide $ telnet pro-linux.de 80
Trying 217.160.186.4...
Connected to pro-linux.de.
Escape character is '^]'.
```

- Dann erst beginnt die eigentliche HTTP-Session:

```
GET / HTTP/1.0
```

  Damit ist unser Befehl abgeschlossen[37] und muss noch mit einer Leerzeile terminiert werden:

  (Leerzeile)

[37]Wir könnten hier, nach dem »GET«, auch noch weitere Headerfelder, wie zum Beispiel bei HTTP/1.1 das »Host«-Feld oder den Referer (auf Seite 56)), unterbringen

- Die Antwort, falls alles geklappt hat, ist dann:

  ```
  HTTP/1.1 200 OK
  ```

  Diese Zeile übermittelt den Statuscode. Hier ist er »200«, was einem »OK« (direkt nachfolgend) entspricht.

  Dann kommen einige Header-Felder, in der gleichen Syntax wie wir sie schon von den Mails her kennen (auf Seite 21).

  – Wann wurde die Seite zuletzt geändert?

    ```
    Date: Wed, 16 Jul 2003 12:56:36 GMT
    ```

  – Welcher Server wird verwendet?

    ```
    Server: Apache/42.3.27
    ```

  – Gibt es Keep-Alive?

    ```
    Connection: close
    ```

  – Was ist der MIME-Type (auf Seite 23) des Inhalts?

    ```
    Content-Type: text/html
    ```

- Dann erst kommt, wieder getrennt durch eine Leerzeile, der Dokument-Inhalt:

  ```
  (Leerzeile)
  <!DOCTYPE(...)
  ```

8.4 HTTP/1.1

Als Minimalanforderung wird bei HTTP/1.1 ein zusätzliches Header-Feld vom Client erwartet, »Host«:

```
GET / HTTP/1.1
Host: linide.sf.net
(Leerzeile)
```

Neu dazugekommen ist also nur der »Host«-Header. Dieser wird von vielen Hosting-Angeboten genutzt, die ihren Kunden nur Webspace, aber keinen eigenen Server. So kann also ein Kunde eine eigene Domain haben, **ohne** dass ein weiterer Server mit einer eigenen IP benötigt wird.

Der Webserver unterscheidet sein Webangebot dann einfach anhand jenes Headerfeldes.

Zu beachten ist auch, dass die Verbindung bei HTTP/1.1 **nicht** nach erfolgtem Reply automatisch geschlossen wird. Dies liegt am bei HTTP/1.1 standardmäßig aktiviertem Keep-Alive (auf Seite 59).

8.5 HTTP-Proxies

HTTP-Proxies sind wirklich cool, wie später im Kapitel über die Automatisierung mit Hilfe von Skripten (auf Seite 117) noch näher erläutert wird.

Proxies übernehmen das Auflösen des (Ziel-)Hostnamens zu einer IP-Adresse (mittels DNS (auf Seite 104)) und verbinden sich dann, stellvertretend für den eigenen Computer, mit dem eigenen Computer.

Die Antwort liefert der Proxy dann an den eigenen Computer zurück.

Auch hier unterscheidet sich das praktische Vorgehen wenig von HTTP/1.0 oder HTTP/1.1, nur dass es jetzt keinen Standardport mehr gibt: Übliche Ports für HTTP-Proxies sind »3128«, »8080« und »79«. Außerdem muss statt nur einem Pfad jetzt die ganze URL geliefert werden.

Um zum Beispiel über die Proxy »proxy.as-netz.de:8080«[38] die Seite »http://linide.sf.net/uebersicht.html« abzurufen, benutzt man:

```
iblech@thestars theguide $ telnet proxy.as-netz.de 8080
Trying 80.81.6.40...
Connected to proxy.a-city.de.
Escape character is '^]'.
GET http://linide.sf.net/uebersicht.html HTTP/1.1
Host: linide.sf.net
(Leerzeile)
```

Die Antwort (samt Statuscodes) ist dann dieselbe wie bei normalem HTTP.

8.6 Überblick

Bis jetzt sollte also folgendes klar sein:

Typ	Standard-Port(s)	Adressenübergabe	»Host«-Angabe
HTTP/1.0	»80«	Nur Pfad	Nein
HTTP/1.1	»80«	Nur Pfad	Ja
HTTP-Proxy	»3128«, »8080«	Volle URL	Je nach Zielprotokoll

[38] Diese Schreibweise (»host:port«) wird oft verwendet, da sie sowohl den Hostnamen als auch den zu verwendenen (TCP-)Port angibt.

8.7 Referer

Aber jetzt wird es erst richtig spanned: Oft will man aus irgendwelchen vollkommen unbekannten Gründen[39] eine Website abrufen. Im Browser funktioniert das absolut unproblematisch, in automatisierten Skripten jedoch nicht. Oft wird man stattdessen zur Homepage weitergeleitet.

Die Magie liegt im sogenannten **Referer**. Der Referer ist ein spezielles Header-Feld, was die URL der zuletzt besuchte Website enthält.

So kann ein Hoster zum Beispiel sehen, von welchen Ländern (Top-Level-Domains), von welchen Unterseiten, usw. kommt. Und er kann so auch prüfen, ob man erst die Homepage „angeklickt" hat, und nicht einen direkten Link benutzt.

Man kann den Referer aber natürlich auch faken. Angenommen, wir wollen via Telnet die Seite »http://www.microsoft.com/unterseite.asp« abrufen, der Seitenbetreiber zwingt uns aber, zuvor auf »http://www.microsoft.com/« gewesen zu sein. Dann würden wir verwenden:

```
GET /unterseite.asp HTTP/1.1
Host: www.microsoft.com
Referer: http://www.microsoft.com/
(Leerzeile)
```

Im Kapitel über die Automatisierung von HTTP (auf Seite 117) werden dazu noch einige Beispiele folgen.

8.8 Weiterleitung

Oft verweist eine Adresse auf eine andere, zum Beispiel verweist »http://sf.net/« auf »http://sourceforge.net/«. Während dies zum einen mit Hilfe eines Meta-Feldes von (X)HTML möglich ist, ist die HTTP-Methode (verfügbar sowohl bei HTTP/1.0 als auch bei HTTP/1.1) wesentlich eleganter, da sie bei Dateien jedes Formats funktioniert, nicht nur bei HTML.

Erkennbar ist so eine Weiterleitung daran, dass statt des Statuscodes »200« »302« zurückgeliefert wird. Im »Location«-Headerfeld wird dann die Zieladresse übermittelt.

Um also auf das Beispiel mit Sourceforge zurückzukommen:

[39] muhahaha. . . ;-)

- Zuerst verbinden wir uns mit Port »80« von »sf.net«, und verlangen ganz normal »/«:

```
iblech@thestars theguide $ <I<C<telnet sf.net 80>>>
Trying 66.35.250.203...
Connected to sf.net.
Escape character is '^]'.
GET / HTTP/1.1
Host: sf.net
(Leerzeile)
```

- Aber als Antwort erhalten wir jetzt »302 Found«:

```
HTTP/1.1 302 Found
```

- Es folgen einige andere Header-Felder. . .

```
Date: Fri, 10 Sep 2004 13:50:22 GMT
Server: Apache/1.3.27 (Unix) PHP/4.3.2 mod_ssl/2.8.12
OpenSSL/0.9.6b
Connection: close
Transfer-Encoding: chunked
Content-Type: text/html
```

- ...und schließlich die Zieladresse. Danach wird die Verbindung geschlossen:

```
Location: http://sourceforge.net/index.php
(Leerzeile)
0
(Leerzeile)
Connection closed by foreign host.
iblech@thestars theguide $
```

8.9 HEAD-Request

Wenn der Inhalt einer Seite vollkommen uninteressant, nur der Header von Bedeutung ist, dann sehe ich keinen Zufall – dann erkenne ich die Bestimmung[40] und verwende den »HEAD«-Request:

```
HEAD / HTTP/1.1
Host: hostname
(Leerzeile)
HTTP/1.1 200 OK
Date: Fri, 18 Jul 2003 19:00:37 GMT
Server: Apache/21.3.19 (GNU Gentoo/Hurd) mod_fastcgi/2.2.2
mod_dtcl
Content-Type: text/html
(Leerzeile)
```

[40] YEAH PHEAR TEH MATRIX

8.10 TRACE-Request

Manchmal möchte man auch wissen, welche Daten ein Server nach passieren aller eventuellen Proxies tatsächlich erhält. Dazu benutzt man den nur bei **HTTP/1.1** vorhandenen »TRACE«-Request:

```
TRACE http://www.linux.org/ HTTP/1.1
Test-Header: Test-Inhalt
(Leerzeile)
HTTP/1.1 200 OK
Date: Wed, 06 Aug 2003 16:46:47 GMT
Server: Apache/1.3.28 (Linux) mod_perl/1.28 \
  mod_ssl/2.8.15 OpenSSL/0.9.7a
Content-Type: message/http
X-Cache: MISS from true
Proxy-Connection: close
(Leerzeile)
TRACE / HTTP/1.1
Cache-Control: max-age=259200
Connection: keep-alive
Host: www.linux.org
Test-Header: Test-Inhalt
Via: 1.0 true:3128 (Squid/2.4.STABLE1)
X-Forwarded-For: 10.0.0.3
(Leerzeile)
```

Im Beispiel geht der Request also über eine Proxy (Squid), der ursprüngliche Absender war »10.0.0.3«[41].

8.11 User-Agent

Manchmal bekommt man beim Surfen auch die Meldung „Tut uns wirklich sehr sehr leid, aber wir optimieren unsere Seiten auf den Microsoft (R)(TM)(C) Internet (R)(TM)(C) Explorer (R)(TM)(C), Sie werden jedoch Mozilla oder eine andere inkompatible Internetznutzungsweichware. Klicken Sie hier, um sich die aktuelle Version des IE herunterzuladen...“. Die Antwort auf die Frage „Woher wissen die eigentlich, welchen Browser ich benutze?“ liegt wieder in einem zusätzlichen Header-Feld, das der Browser dem Server sendet: »User-Agent«:

[41] Natürlich können alle Header gefälscht sein – Merke: Traue nichts, was nicht signiert ist!

```
GET / HTTP/1.1
Host: microsoft.com
User-Agent: Mozilla/5.0 (compatible; Konqueror/3.1; Linux)
(Leerzeile)
HTTP/1.1 200 OK
Server: Microsoft IIS Bug-free(TM)
Content-Type: text/plain
(Leerzeile)
Sorry, aber diese Seite...
```

8.12 Accept-Language

Ein Seitenbetrieber kann die vom Client mitgeschickten Header aber auch für gute Zwecke nutzen: Verlangt man zum Beispiel »http://debian.org/«, so bekommt man (je nach Browser und Einstellungen) automatisch die deutsche Version angezeigt. Diese Information erhält der Server über das »Accept-Language«-Headerfeld:

```
GET / HTTP/1.1
Host: debian.org
Accept-Language: de,en
(Leerzeile)
HTTP/1.1 200 OK
Content-Type: text/plain
(Leerzeile)
(Deutsche Version...)
```

Üblicherweise werden mehrere von der ISO normierten Sprachen-Kürzel mit einem »,« (Komma) getrennt.

8.13 Keep-Alive

Mit Keep-Alive wird die Verbindung zum Server auch nach einem Request (samt Reply) offen gehalten. Dies ist besonders bei HTML-Seiten mit vielen Bildern sehr nützlich, da das Verbinden mit dem Server oftmals der längste Teil einer HTTP-Transaktion ist. Bei einer Seite mit 20 Bildern wären 21 Verbindungen erforderlich!

Bei HTTP/1.1 ist Keep-Alive standardmäßig aktiviert. Möchte man die Verbindung nach einem Request schließen, so muss man bei dem letzten Request explizit den »Connection«-Header benutzen:

```
Connection: close
```

Ein letzter Request könnte also zum Beispiel so aussehen:

```
GET /letztes-bild.png HTTP/1.1
Host: der-server.org
Connection: Close
(Leerzeile)
```

Möchte man bei HTTP/1.0 Keep-Alive aktivieren, so kann man

```
Connection: Keep-Alive
```

benutzen. Allerdings ist Keep-Alive bei HTTP/1.0 im Zusammenhang mit Proxies nicht zu empfehlen: Wenn die Proxy nicht aktiviertes Keep-Alive erkennt, so kann sie auch nicht das Ende einer Übertragung erkennen. Also würde der Proxy unbegrenzt lange (=bis der Timeout eintritt) auf die Antwort des Servers warten. . .

Jetzt gibt es allerdings en großes Problem: Wenn die Verbindung nicht geschlossen wird, woher weiß dann ein Client, dass die Seite fertig übertragen wurde? Eine einfache Antwort bietet der »Content-Length«-Header, der die Anzahl der Bytes der Übertragung angibt[42].:

```
HTTP/1.1 200 OK
Content-Length: 12
(Leerzeile)
Hallo, Welt!
```

8.13.1 Chunked-Encoding

Bei (z.B.) CGI-Skripten kann der Server aber nicht wissen, wir groß die Seite ist, sie wird ja dynamisch generiert. Aber auch dazu gibt es eine Lösung: Chunked-Encoding. Bei diesem Verfahren wird eine Antwort in viele kleinere Teile zerlegt, wobei bei jedem kleineren Teil die »Content-Length« bekannt ist:

---

[42]Im Beispiel ist die Zeile übrigens **nicht** nicht mit EOL terminiert.

```
GET / HTTP/1.1
Host: somehost
(Leerzeile)
HTTP/1.1 200 OK
Date: Thu, 07 Aug 2003 08:31:29 GMT
Transfer-Encoding: chunked
Content-Type: text/html
(Leerzeile)
ea1
(Jetzt folgen 0xea1 (=3745 dezimal) Bytes)
f7
(Jetzt folgen nochmal 0xf7 (=247 dezimal) Bytes)
0
(Leerzeile)
```

Auch kann ein Client, zum Beispiel bei »POST« (auf Seite 65), Chunked-Encoding verwenden.

Laut RFC 2616[43] muss jeder HTTP/1.1 Client und Server Chunked-Encoding unterstützen.

8.14 Partial Content

Manchmal möchte man nur einen bestimmten Bereich einer Datei herunterladen, zum Beispiel wenn ein Download abgebrochen wurde und dann später erneut aufgenommen werden soll. Dazu gibt es bei HTTP/1.1 den »Range«-Header:

```
GET /proxy.log HTTP/1.1
Host: mars
Range: bytes=17-42
(Leerzeile)
HTTP/1.1 200 OK
Content-Type: text/plain
(Leerzeile)
HTTP/1.1 206 Partial Content
Accept-Ranges: bytes
Content-Length: 26
Content-Range: bytes 17-42/1158
Content-Type: text/plain
(Leerzeile)
(Bytes 17 bis 42 von /proxy.log)
```

[43] http://www.faqs.org/rfcs/rfc2616.html

8.15 Cookies

Cookies, definiert in RFC 2109[44], können kleinere Informationen über den Benutzer speichern, zum Beispiel welche Waren sich in seinem E-Warenkorb seines E-Shops befinden.

Auch können sie jedem Benutzer eine von der IP unabhängige ID zuweisen, um jeden Benutzer bei einem späteren Besuch wiedererkennen zu können.

Cookies werden[45] vom Server durch ein Headerfeld gesetzt (im Beispiel ein verbessertes Google):

```
GET / HTTP/1.1
Host: www.google.de
(Leerzeile)
HTTP/1.1 200 OK
Date: Fri, 01 Aug 2003 09:17:00 GMT
Content-Type: text/html
Set-Cookie: PREF="ID=450b1d25610"; \
  expires="Sun, 17-Jan-2038 19:14:07 GMT"; Path="/"; \
  Domain=".google.de"; Version="1"
(Leerzeile)
<html><head>...
```

Durch den »Set-Cookie«-Header haben wir ein Cookie erhalten. Die Variablen des Cookies sind:

»PREF=...«
: Dadurch kann Google uns wieder erkennen und unsere Einstellungen (Sprache, etc.) speichern.

»expires=...«
: Mit diesem Parameter legt Google fest, wann das Cookie verfällt, das heißt wann die gesetzten Einstellungen nicht mehr gültig sind[46].

---

[44] http://www.faqs.org/rfcs/rfc2109.html

[45] von JavaScript-Idiotie einmal abgesehen. . .

[46] Wen es jetzt wundert, warum das Cookie am 17. Januar 2038 („krumme“ Zahl) abläuft und nicht zum Beispiel am 1. Januar 2010 („gerade“ Zahl), der muss wissen, wie Unix intern seine Uhr verwaltet: Unix zählt die Anzahl der Sekunden seit dem 1. Januar 1970, 0 Uhr GMT. Diese Anzahl hält Unix in einem (noch) 32-Bit breitem Register fest. Und die damit höchste darstellbare Zahl, 2147483647, zurück in „unser“ System umgerechnet, ist genau der 17. Januar 2038. Eine Sekunde später ist für Unix wieder der 1. Januar 1970.

»**path=...**«, »**domain=...**«
Diese Variable bestimmt den Zugriff auf den Cookie: Nur alle Dateien, ab »/« (in dem Fall also alle), der Domain »irgendwas.google.de« haben Zugriff auf das Cookie. »www.altavista.de« darf also nicht den Inhalt des Cookies erhalten.

Bei jedem weiteren Seitenabruf müssen alle für die jeweilige Seite relevanten (also nicht verfallene Cookies) Cookies mit dem »Cookie«-Headerfeld übertragen werden, und zwar genau so, wie man sie erhalten hat. Auch muss natürlich auf die Zugriffsbestimmungen geachtet werden.

Ein zweiter Request auf »http://www.google.de/« muss dann also so aus sehen:

```
GET / HTTP/1.1
Host: www.google.de
Cookie: PREF="ID=450b1d25610"; \
  expires="Sun, 17-Jan-2038 19:14:07 GMT"; $Path="/"; \
  $Domain=".google.de"; $Version="1"
(Leerzeile)
HTTP/1.1 200 OK
Date: Fri, 01 Aug 2003 09:23:13 GMT
Content-Type: text/html
(Leerzeile)
<html><head>...
```

Besonderen Variablen, wie »Path«, »Domain« und »Version«, muss dabei ein »$« (Dollarzeichen) voranstehen. Die „normalen“ Werte (»PREF«, »expires«) werden „roh“ gesendet.

Hier fällt auch auf, dass der Server nicht ein neues Cookie setzen will (»Set-Cookie«-Header fehlt). Das bedeutet also, dass Google uns nun „kennt“. Würden wir eine Einstellung verändern (Sprache, etc.) würden wir ein neues Cookie erhalten. Wenn wir dieses veränderte Cookie nicht mitschicken würden, so kämen wieder Googles Default-Einstellungen zum Einsatz.

8.16 Formulare

Oft sind auf einer Website Formulare zu finden, um zum Beispiel bei einer Umfrage abzustimmen oder dem Webmaster eine Nachricht zukommen zulassen. In XHTML realisiert sieht das zum Beispiel so aus:

```
<form action="http://host/umfrage.p6" method="get">
  Name: <input type="text" name="name" /> <br />
  Alter: <input type="text" name="alter" /> <br />
  OS: <input type="text" name="os" /> <br />
  <input type="submit" value="Abschicken!" />
</form>
```

Die Übertragung solcher Formulare ist über zwei HTTP-Requests möglich: Den einfachen »GET«-Request, bei dem allerdings die maximale Länge der Formulardaten (eigentlich der gesamten URL) nicht mehr als 1024 Zeichen betragen darf, und den etwas schwieriger verwendbaren »POST«-Request.

Im Abschnitt über Telnet (auf Seite 15) habe ich bemerkt, dass viele Leute Telnet als unsicher abstempeln, weil Passwörter im Klartext über´s Netz gehen. Jetzt frage ich diese Leute: Wieso ist nicht HTTP genauso unsicher? Auch bei HTTP gehen **alle Formulardaten** unverschlüsselt über´s Netz. HTTP ist also genauso unsicher wie Telnet, POP3, IMAP4, NNTP, FTP und IRC. Möchte man Sicherheit, muss man die HTTP-Verbindung (oder jede andere Verbindung) mit z.B. SSL/TLS (Secure Socket Layer/Transport Layer Security) verschlüsseln. Im Browser wird eine mit SSL verschlüsselte HTTP-Verbindung als Protokoll »https« angezeigt, also z.B. »https://online.shop/« statt »http://online.shop/«.

8.16.1 GET-Request

Der »GET«-Request ist ja schon vom „normalen" Abrufen von Websites bekannt, das heißt: Nichts neues Lernen!

Bei »GET« werden die Feldnamen von den Feldinhalten mit einem »=« (Gleichheitszeichen) getrennt, und dann an die URL, getrennt mit einem »?« (Fragezeichen), angehängt. Die einzelnen Parameter unter sich werden mit einem »&« (Ampersand) getrennt.

Z.B. wird aus dem Wert »Ingo« für »name«, »16« für »alter« und »Gentoo« für »OS« im Browser folgende URL konsturiert:

```
http://host/umfrage.p6?name=Ingo&alter=16&os=Gentoo
```

Zusammenfassend wird

- die „normale" URL von den Parametern mit einem »?«,

- jeder Parameter von den anderen mit einem »&« und
- Feldname von Feldinhalt mit einem »=« getrennt.

Auch hierzu mehr im Kapitel über die Automatisierung der ganzen Vorgänge (auf Seite 117).

8.16.2 POST-Request

Beim »POST«-Request werden die Felder nicht in der URL gespeichert, sondern im HTTP-Request selbst. Dies ermöglicht Übertragungen größer als einem Kilobyte.

Unser Beispielrequest sieht bei Verwendung von POST so aus:

```
POST /umfrage.p6 HTTP/1.1
Host: host
Content-Type: application/x-www-form-urlencoded
Content-Length: 29
(Leerzeile)
name=Ingo&alter=15&os=Gentoo
(Leerzeile)
HTTP/1.1 200 OK
Date: Fri, 01 Aug 2003 10:17:48 GMT
Content-Type: text/html
(Leerzeile)
<html>(...)
```

Der Request ist also verlängert, nach der Leerzeile kommt **nicht** (wie etwa bei »GET« und »HEAD«) die Antwort, sondern der Inhalt des Formulars.

Wichtig ist auch der »Content-Length«-Header. Die Länge muss exakt mit der Anzahl der Zeichen der Übertragung übereinstimmen. Kennt man die Länge nicht im Vorraus, kann man auch hier Chunked-Encoding verwenden:

```
POST /umfrage.p6 HTTP/1.1
Host: host
Content-Type: application/x-www-form-urlencoded
Transfer-Encoding: chunked
(Leerzeile)
a
name=Ingo&
12
alter=15&os=Gentoo
0
(Leerzeile)
```

8.17 Requestübersicht

Request	Wirkung
»GET /pfad«	Holt die angegebene Datei
»GET /pfad?feld1=name1&feld2=name2«	Sendet Parameter via »GET«
»POST /pfad«	POSTed Daten an angegebenen Pfad
»HEAD /pfad«	Holt nur die Seiten-Header (»Content-Type«, etc.)
»TRACE /pfad«	Echoed den gesamten Request (für´s Debugging)

Typische Header-Felder sind:

Header-Feld	Bedeutung	Wird gesendet vom
»Host«	Hostname des Servers	Client
»Connection«	Keep-Alive benutzen?	Client und Server
»Referer«	Zuletzt besuchte Seite	Client
»User-Agent«	Verwendeter Browser	Client
»Cookie«	Gesetzte Cookies	Client
»Accept-Language«	Gewünschte Sprachen	Client
»Server«	Benutzte Server-Software	Server
»Location«	Weiterleitungsziel	Server
»Set-Cookie«	Cookie(s) setzen	Server
»Content-Type«	MIME-Typ der vom Client angeforderten Seite/der Daten, die gePOSTed werden	Client und Server

8.17.1 Beispielsitzung

Als Beispiel hier nochmal eine komplette Sitzung, unter Verwendung der meisten Befehle und Header-Felder.

```
POST /cgi-bin/login.pl HTTP/1.1
Host: wewewe.mein-loginserver.de
Connection: close
Referer: http://wewewe.mein-loginserver.de/
User-Agent: Lynx/2.8.4rel1 (full-comatible; 128Bit GNU/Linux)
Cookie: $Path="/"; Domain="wewewe.mein-loginserver.de";
        $Version="1"; ColorScheme="default"
Accept-Language: de
Content-Type: application/x-www-form-urlencoded
Content-Length: 29
(Leerzeile)
username=iblech&password=2600
(Leerzeile)
HTTP/1.1 302 Found
Date: Fri, 18 Jul 2003 19:00:37 GMT
Server: Apache/21.3.19 (GNU Gentoo/Hurd)
Content-Type: text/plain
Location: http://wewewe.mein-loginserver.de/cgi-bin/main.pl
Set-Cookie: Path="/"; Domain="wewewe.mein-loginserver.de";
            $Version="1"; logged_in="true"; username="iblech";
            ColorScheme="red";
            SID="d17ed26ab2c53f52a91777fcc5348849"
(Leerzeile)
Eine Weiterleitung erfolgt...
```

9 Gopher

Gopher, definiert in RFC 1436[47] ist, oder besser war, ähnlich wie HTTP früher, ein Protokoll zum Abrufen von Informationen. Leider stirbt der „Gopherspace“ immer mehr aus, da den Lamerz und D00dz c00le Flash-Anim4tionz wichtiger sind als das schnelle Auffinden von Dateien. . . :((

Trotzdem werde ich hier Gopher beschreiben.

[47] http://www.faqs.org/rfcs/rfc1436.html

9.1 Design

Die Informationen sind bei Gopher in einer Art Dateibaum hierarchisch angeordnet, wobei es keine Rolle spielt, auf welchem Server die Daten wirklich vorliegen. Navigiert wird mittels Menüs. Durch den Verzicht auf zwar schön anzusehende, aber nicht funktionale Design, war es möglich, nur durch wenige Verzweigungen, also Auswählen von Menüeinträgen, zum Ziel zu kommen.

9.2 Technik

Bei Gopher gibt es einige verschiedene Dateitypen, die durch eine Ziffer oder einen Buchstaben symbolisiert werden:

Kennzeichen	Typ
»0«	ASCII-Datei
»1«	Verzeichnis
»3«	Fehler
»6«	UU-enkodierte Datei
»8«	Link auf eine Telnet-Sitzung
»9«	Binäre Datei
»h«	HTML-Seite
»i«	Nichtauswählbarer Menü-Eintrag (z.B. eine Willkommensmeldung)
»P«	PDF-Dokument

Die Pfadinformation von Gopher-URLs (auf Seite 53) setzt sich immer aus dem Dateityp und dem Pfad zusammen, zum Beispiel »1/GNU«, »0/GNU/GPL.txt« oder »h/GNU/GPL.html«.

9.3 Beispielsitzung

Wir werden zuerst die Einstiegsseite von »quux.org« holen. »70« ist der Standard-Port von Gopher, also geben wir ein:

```
iblech@thestars theguide $ telnet quux.org 70
Trying 69.10.152.57...
Connected to www.quux.org.
Escape character is '^]'.
```

```
(Leerzeile)
iWelcome to...       fake        (NULL)   0
i                    fake        (NULL)   0
iThis server...,     fake        (NULL)   0
ifunny, or just...   fake        (NULL)   0
iThere are many...   fake        (NULL)   0
i                    fake        (NULL)   0
0About This Server   /About.txt  quux.org 70    +
1Archives            /Archives   quux.org 70    +
1Books               /Books      quux.org 70    +
(...)
.
Connection closed by foreign host.
iblech@thestars theguide $
```

Die einzelnen Rückgabefelder sind dabei durch einen Tabulator getrennt, und geben der Reihe nach

- den Link-Namen (mit vorrangestellten Dateityp-Kennzeichen),
- den Zielpfad,
- den Server des Ziels
- und dessen Port

an. Das Ende der Übertragung wird wie so oft mit einem einzelnen Punkt auf einer Zeile markiert, woraufhin die Verbindung auch geschlossen wird. „Keep-Alive“ wie bei HTTP/1.1 (auf Seite 59) ist für Gopher also ein Fremdwort. Bei der Übertragung von Dateien wird **kein** abschließender Punkt mitgeschickt.

In einem Client (im Beispiel Lynx) sieht das Einstiegsmenü so aus:

```
 ----------------------------------------------
|                                              |
|                        Gophermenü (p1 of 6)|
|                  Gophermenü                  |
|                                              |
|      Welcome to gopher at quux.org!          |
|      This server has a lot of information of h|
|istoric interest,                             |
|       funny, or just plain entertaining -- all|
|presented in Gopher.                          |
|       There are many mirrors here of rare or v|
|-- Leertaste für mehr, Pfeiltasten zum Bewegen |
|----------------------------------------------|
```

Ist das Ziel unserer Anfrage eine Datei, ist das auch kein Problem:

```
iblech@thestars theguide $ telnet quux.org 70
Trying 69.10.152.57...
Connected to www.quux.org.
Escape character is '^]'.
/About This Server.txt
Welcome to the gopher server at quux.org!

This is one of the world's few maintained, modern
gopher servers.  On it, you will find a huge
collection of information, files, software,
archives, and mirrors pertaining to a wide variety
of topics of interest to computer enthusiasts.
You'll find humor, jargon files, ZCode games and
interpreters, decompressors for both modern and
old compression formats, emulators for old CPUs,
and even the occasional old operating system.
(...)
Connection closed by foreign host.
iblech@thestars theguide $
```

9.4 Abschluss

So ein geniales Protokoll, wo die Dateien in einem Dateisystem organisiert sind, sollte, meiner Meinung nach, auch weiterhin unterstützt werden.

Gopher lebt!

ahem

10 FTP

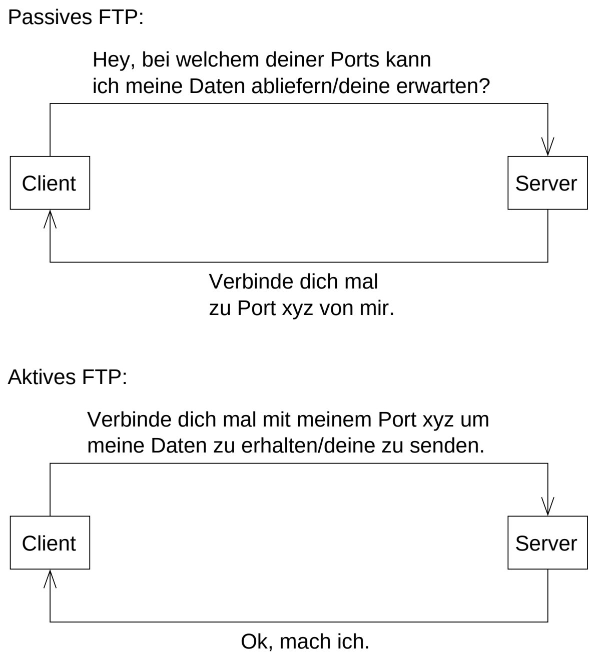
Das File Transfer Protocol, 1971 ursprünglich definiert in RFC 114[48], beschreibt eine einfache Möglichkeit des Datenaustauschs. Wie eigentlich alle „alten" Protokolle basiert es auf ASCII, ist allerdings nicht so leicht zu bedienen wie zum Beispiel POP3.

[48] http://www.faqs.org/rfcs/rfc114.html

10.1 Design

Bei FTP ist immer eine Kontrollverbindung (standardmäßig zu TCP-Port »21«) offen. Dazu kommt dann noch eine Datenverbindung, die erst später geöffnet wird. Der Port dieser Verbindung wird erst durch die Kontrollverbindung festgelegt.

Die Datenverbindung kann dabei sowohl vom Client (passives FTP) als auch vom Server aus (aktives FTP) geöffnet werden:



Man kann folglich FTP als eine TCP-Proxy (!) benutzen, sofern aktives FTP zugelassen ist, was sogar bei <<ftp.microsoft.com>> der Fall ist. . .

10.2 Passives FTP

Um irgendeine Transaktion ausführen zu können (Verzeichnislisting erhalten, Dateien herunter- oder hochladen, Verzeichnisse erstellen, etc.) müssen wir uns zuerst einloggen. FTPs Standard-Port für den Kontrollkanal ist TCP-Port »21«:

```
iblech@thestars theguide $ telnet ftp.microsoft.com 21
```

```
Trying 207.46.133.140...
Connected to ftp.microsoft.com.
Escape character is '^]'.
220 Microsoft FTP Service
USER anonymous
331 Anonymous access allowed, send identity (e-mail \
  name) as password.
PASS abuse@aol.com
230-This is FTP.Microsoft.Com.
230 Anonymous user logged in.
```

Hier kann man bereits eine Ähnlichkeit zu POP3 und NNTP feststellen: Es werden auch hier Statuscodes verwendet. Der »anonymous«-Benutzer ist auf den meisten Servern vorhanden, natürlich ohne Schreibrechte. Als Passwort wird oft die E-Mail-Adresse verlangt, damit der Betreiber der Site uns kontaktieren (=Spam versenden) kann.

Jetzt kommt der entscheidene Befehl von passivem FTP, der nach jeder Übertragung wieder neu eingegeben werden muss:

```
PASV
227 Entering Passive Mode (207,46,133,140,191,242).
```

Diese wirre Zahlenkombination verrät uns, wohin wir eine TCP-Verbindung aufbauen müssen. Die IP ist klar in den ersten vier Blöcken angegeben (»207.46.133.140«). Der Port errechnet sich aus $a * 256 + b$, wobei a der ersten Zahl nach der IP entspricht (»191«) und b der zweiten (»242«). Also müssen wir eine Verbindung zu »207.46.133.140:49138« aufbauen. Dies erledigen wir in einer zweiten Shell (auf schlechten Systemen „MS-DOS-Eingabeaufforderung“ genannt):

```
iblech@thestars theguide $ telnet 207.46.133.140 49138
Trying 207.46.133.140...
Connected to 207.46.133.140.
Escape character is '^]'.
```

Jetzt kann mit Befehlen (auf Seite 76) der Gegenstand der Übertragung genannt werden.

Kontrollterminal:

```
iblech@thestars theguide $ telnet ftp.microsoft.com ftp
Trying 207.46.133.140...
Connected to ftp.microsoft.com.
Escape character is '^]'.
220 Microsoft FTP Service
USER anonymous
331 Anonymous access allowed, send identity (e-mail \
  name) as password.
PASS abuse@aol.com
230-This is FTP.Microsoft.Com.
230 Anonymous user logged in.
PASV
227 Entering Passive Mode (207,46,133,140,191,242).
LIST
125 Data connection already open; Transfer starting.
226 Transfer complete.
QUIT
221 Thank-you for using Microsoft products!
Connection closed by foreign host.
iblech@thestars theguide $
```

Datenterminal:

```
iblech@thestars theguide $ telnet 207.46.133.140 49138
Trying 207.46.133.140...
Connected to 207.46.133.140.
Escape character is '^]'.
dr-xr-xr-x   1 owner    group       0 Nov 25  2002 bussys
dr-xr-xr-x   1 owner    group       0 May 21  2001 deskapps
dr-xr-xr-x   1 owner    group       0 Apr 20  2001 developr
dr-xr-xr-x   1 owner    group       0 Nov 18  2002 KBHelp
dr-xr-xr-x   1 owner    group       0 Jul  2  2002 MISC
dr-xr-xr-x   1 owner    group       0 Dec 16  2002 MISC1
dr-xr-xr-x   1 owner    group       0 Feb 25  2000 peropsys
dr-xr-xr-x   1 owner    group       0 Jan  2  2001 Products
dr-xr-xr-x   1 owner    group       0 Apr  4 13:54 PSS
dr-xr-xr-x   1 owner    group       0 Sep 21  2000 ResKit
dr-xr-xr-x   1 owner    group       0 Feb 25  2000 Services
dr-xr-xr-x   1 owner    group       0 Feb 25  2000 Softlib
Connection closed by foreign host.
iblech@thestars theguide $
```

10.3 Aktives FTP

Richtig interessant ist allerdings **aktives** FTP, da wir selbst bestimmen können, wohin der Server seine Datenverbindung aufbauen soll!

Die Basis ist die gleiche wie bei passivem FTP, also erstmal verbinden und authentifizieren:

```
iblech@thestars theguide $ telnet ftp.microsoft.com 21
Trying 207.46.133.140...
Connected to ftp.microsoft.com.
Escape character is '^]'.
220 Microsoft FTP Service
USER anonymous
331 Anonymous access allowed, send identity (e-mail \
  name) as password.
PASS abuse@aol.com
230-This is FTP.Microsoft.Com.
230 Anonymous user logged in.
```

Nun benutzen wir den »PORT«-Befehl, um dem Server mitzuteilen, wohin er seine Daten schicken soll bzw. unsere Daten erhalten soll. Die Syntax ist dabei die gleiche wir bei »PASV«, nur dass wir jetzt selbst die Parameter festlegen. Angenommen, unsere IP-Adresse ist »80.81.9.189« und unser gewünschter Port ist »4712«, so geben wir ein:

```
PORT 80,81,9,189,18,104
200 PORT command successful.
```

ein. Die Zahlen, die den Port spezifizieren errechnen sich dabei durch die Formeln $a = \mathrm{int}\left(\frac{p}{256}\right)$ und $b = p\%256$.

In einer anderen Konsole lauschen wir auf dem Port, um die Ausgabe des Servers (Verzeichnislisting oder Herunterladen von Dateien) bzw. unsere Daten einzutippen:

```
iblech@mars iblech $ nc -vlp 4712
listening on [any] 4712 ...
```

Nun kann (wieder im Datenkanal) die Übertragung zum Beispiel mit »LIST«, »RETR dateiname« oder »STOR dateiname« initiiert werden.

Kontrollterminal:

```
iblech@thestars theguide $ telnet ftp.microsoft.com ftp
Trying 207.46.133.140...
Connected to ftp.microsoft.com.
Escape character is '^]'.
220 Microsoft FTP Service
USER anonymous
331 Anonymous access allowed, send identity (e-mail \
  name) as password.
PASS abuse@aol.com
230-This is FTP.Microsoft.Com.
230 Anonymous user logged in.
PORT 80,81,9,189,15,111
200 PORT command successful.
LIST
125 Data connection already open; Transfer starting.
226 Transfer complete.
QUIT
221 Thank-you for using Microsoft products!
Connection closed by foreign host.
iblech@thestars theguide $
```

Datenterminal:

```
iblech@mars iblech $ nc -vlp 3951
listening on [any] 3951 ...
dr-xr-xr-x   1 owner    group        0 Nov 25  2002 bussys
dr-xr-xr-x   1 owner    group        0 May 21  2001 deskapps
dr-xr-xr-x   1 owner    group        0 Apr 20  2001 developr
dr-xr-xr-x   1 owner    group        0 Nov 18  2002 KBHelp
dr-xr-xr-x   1 owner    group        0 Jul  2  2002 MISC
dr-xr-xr-x   1 owner    group        0 Dec 16  2002 MISC1
dr-xr-xr-x   1 owner    group        0 Feb 25  2000 peropsys
dr-xr-xr-x   1 owner    group        0 Jan  2  2001 Products
dr-xr-xr-x   1 owner    group        0 Apr  4  2003 PSS
dr-xr-xr-x   1 owner    group        0 Sep 21  2000 ResKit
dr-xr-xr-x   1 owner    group        0 Feb 25  2000 Services
dr-xr-xr-x   1 owner    group        0 Feb 25  2000 Softlib
Connection closed by foreign host.
iblech@mars iblech $
```

10.4 Befehlsübersicht

Mit einer Übertragung kann erst begonnen werden, wenn der Datenkanal offen ist.

Befehl	Wirkung	Datenkanal benötigt
»USER username«	Einloggen	Nein
»PASS p455w0rd«	Einloggen	Nein
»QUIT«	Ausloggen	Nein
»LIST«	Listing des aktuellen Verzeichnisses herunterladen	Ja
»RETR dateiname«	»dateiname« herunterladen	Ja
»STOR dateiname«	Datenkanal als »dateiname« speichern	Ja
»APPE dateiname«	Datenkanal an »dateiname« anhängen	Ja
»DELE dateiname«	»dateiname« löschen	Nein
»PWD«	Aktuelles Verzeichnis anzeigen	Nein
»CWD verzeichnis«	In »verzeichnis« wechseln	Nein
»MKD verzeichnis«	»verzeichnis« anlegen	Nein
»RMD verzeichnis«	»verzeichnis« löschen	Nein
»RNFR altername«	Datei oder Verzeichnis »altername« zum Umbennen markieren	Nein
»RNTO neuername«	Markierte Datei in »neuername« umbennen	Nein

10.5 Probleme mit Firewalls

Leider führt FTP oft zu Problemen mit Firewalls: Oft konfiguriert man seine Firewalls so, dass sie **nur akzeptieren, was explizit**

erlaubt wurde (und nicht umgekehrt, „nur verbieten, was explizit verboten wurde, den Rest erlauben"). Nun kann man schön einstellen, dass eingehender Traffic auf TCP-Port »21« erlaubt werden soll. Damit hat der **Kontrollkanal** keine Probleme. Aber die Ports des **Datenkanals** werden ja dynamisch ausgehandelt – und dazu hat die Firewall keine passende Regel, mit dem Ergebnis, das jede Datenübertragung fehlschlägt.

Um dieses Manko zu umgehen gibt es mehrere Möglichkeiten: Zum einen kann man einfach die Firewall-Policy zu „alles erlauben, was nicht explizit verboten wurde" ändern. Schlecht. Oder man kann den FTP-Daemon auf aktives FTP beschränken. Diese Möglichkeit funktioniert, wenn man lokalen Programmen die Verbindungsaufnahme zu beliebigen nicht-lokalen (TCP-)Ports gestattet. Aber auch aktives FTP hat Sicherheitsrisiken (auf der nächsten Seite).

Es gibt aber auch eine sehr viel bessere Möglichkeit, dieses Problem zu beheben: „Intelligente" Firewalls lesen den Kontrollkanal mit und schalten automatisch die ausgehandelten Ports frei. Das passende Kernel-Modul für Linux´ Netfilter (iptables) ist »conntrack_ftp«.

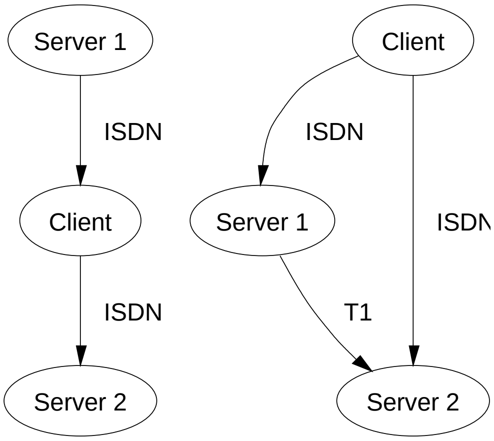
10.6 Goodies

Zwei Features von aktivem FTP machen es jedoch erst richtig interessant: Zum einen kann man eine Datei von einem Server zu einem anderen übertragen, **ohne** die Datei lokal zwischenspeichern zu müssen, und zum anderen kann man aktives FTP als TCP-Proxy (!) benutzen, man kann also, ausgehend von einem FTP-Server, eine Verbindung zu jedem Host auf (fast) jeden Port herstellen!

10.6.1 Übertragung zwischen zwei Servern

Zum einen kann man natürlich eine Datei eines Servers auf einen anderen mit der „konventionellen" Methode übertragen (Datei vom ersten Server auf den eigenen Computer herunterladen und vom eigenen Computer aus dann auf den zweiten Server hochladen). Dass diese Methode sehr viel Zeit in Anspruch nimmt, ist klar, schließlich müssen zweimal die langsamen ISDN-Leitungen verwendet werden.

Mit einer Kombination aus aktivem und passivem FTP kann der eigentliche Datenaustausch auch direkt zwischen den beiden Servern stattfinden: Man lässt sich einfach vom zweiten Server via passivem FTP eine Adresse geben, welche man dann dem ersten Server via aktiven FTP mitteilt. Dann initiiert man auf beiden Servern die Übertragung, zum Beispiel mit »RETR dateiname« auf dem ersten und »STOR dateiname« auf dem zweiten Rechner. Das Ergebnis: Man hat soeben mit T1-Geschwindigkeit (bzw. der Geschwindigkeit, die zwischen den beiden Servern möglich ist) eine Datei übertragen:



Auch die Implementation ist sehr einfach:

Auf Server 2: **`PASV`**
Auf Server 2: `227 Entering Passive Mode (a,b,c,d,e,f).`
Auf Server 1: **`PORT a,b,c,d,e,f`**
Auf Server 1: `200 PORT command successful.`
Auf Server 1: **`RETR file`**
Auf Server 2: **`STOR file`**

10.7 Application-Level-Proxy

Am coolsten ist FTP jedoch, wenn man es als TCP-Proxy einsetzt. Leider bekommt man Antworten nicht zurück, besser wäre also der Begriff „Application-Level-TCP-Port-In-Eine-Richtung-Forwarder" ;). Eine TCP-Proxy ist eine Proxy auf TCP-Ebene.

Auch die Realisierung ist sehr einfach, wenn man auf dem FTP-Server, der aktives FTP unterstützen muss, Schreibrechte besitzt:

- Zuerst speichern wir auf dem FTP-Server (manuell oder via FTP-Programm) eine Datei, die die Komanndos enthält, die dann auf dem Zielrechner ausgeführt werden sollen. Da wir im Beispiel versuchen werden, eine Website abzurufen, verwenden wir:

```
GET / HTTP/1.1
Host: website.de
User-Agent: 1337 FTP
(Leerzeile)
```

- Dann stellen wir die Verbindung her:

```
PORT ip,der,website,de,0,80
200 PORT command successful.
RETR datei-wo-wir-die-http-kommandos-gespeichert-haben
```

- Und fertig ist der HTTP-Request ;-).

Der Nachteil ist natürlich deutlich, man empfängt die Antwort des Zielservers nicht. Aber oft ist das auch egal, zum Beispiel wenn die Zielseite ein Counter ist (der dann natürlich hochzählt) oder eine Abstimmung. . .

10.8 Abschluss

Einige werden sich jetzt fragen, wozu das ganze denn gut sein soll. Aber FTP-via-Telnet nützt nicht nur, wenn man mal eben eine TCP-Proxy brauch, sondern auch, wenn ein FTP-Server defekt ist. Durch die (meist englischen) Fehlermeldungen kann ein Mensch sehr schnell die Ursache des Problems feststellen und so trotzdem die vielleicht sehr wichtigen Dateien transferieren.

11 IRC

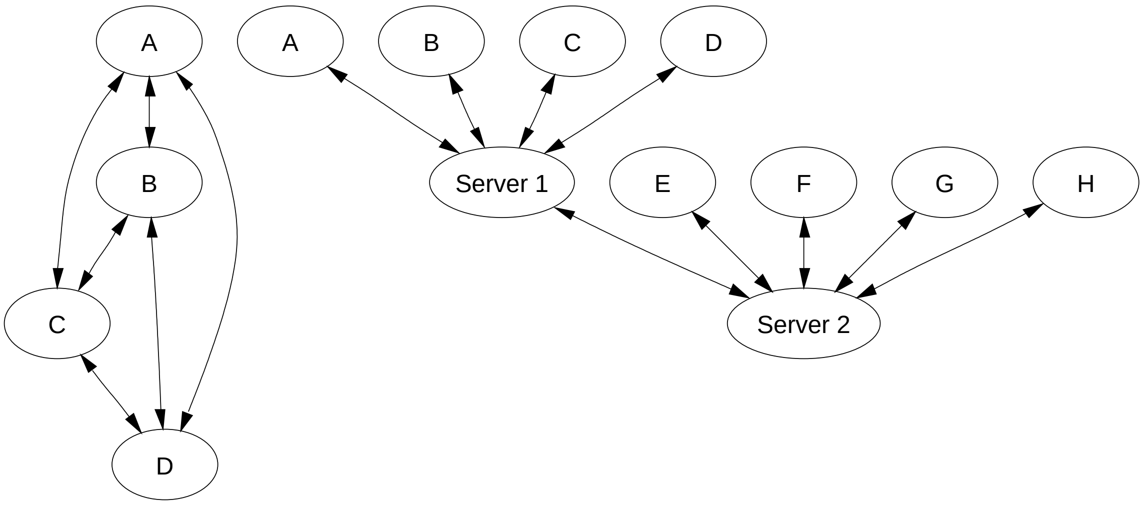
Nun wieder zu einem einfacheren Protokoll: IRC, Internet Relay Chat, ursprünglich definiert in RFC 1459[49]. IRC ist sehr viel jünger als zum Beispiel SMTP oder FTP und übernimmt einen Bereich des Internets, der immer mehr an Zustimmung findet: Echt-Zeit[50] Chatten.

[49] http://www.faqs.org/rfcs/rfc1459.html
[50] mehr oder weniger. . . ;)

11.1 Design

Bei IRC werden Leute mit gleichen Interessen in „Channels“ eingeteilt[51], zum Beispiel »#linux«, »#hurd« oder »#perl«.

Die Kommunikation läuft dann immer über den Server, das heißt, es werden keine Verbindungen zwischen den Client-Rechnern selbst hergestellt. Bei einigen hundert Leuten pro Channel würde man, wenn alle Clients miteinander verbunden werden sollten, tausende Verbindungen benötigen. Wenn sich alle Clients nur mit einem Server verbinden sind nur soviel Verbindungen nötig wie Clients vorhanden sind (Verbindungen zwischen Servern sind natürlich auch noch notwendig).



Dabei tauschen sich die einzelnen IRC-Server eines IRC-Netzwerkes (es gibt meistens mehrere) auch gegenseitig aus, um ein möglichst synchrones IRC-Netz zu gewährleisten. Somit ist es egal, zu welchem Server eines Netzes ein Client sich verbindet: Er sieht trotzdem alle ihn betreffenden Nachrichten aller anderen Clients desselben Netzes.

11.2 Beispielsitzung

Im Folgenden werden wir eine Runde im Channel »#infothek« des Servers »irc.freenode.net:6667« chatten. Dazu verbinden wir uns

[51] natürlich freiwillig. . .

zuerst mit dem Server, woraufhin wir gleich eine Authentifizierungsmeldungen erhalten:

```
iblech@thestars theguide $ telnet irc.gnu.org 6667
Trying 217.172.187.182...
Connected to irc.freenode.net.
Escape character is '^]'.
NOTICE AUTH :*** Looking up your hostname...
NOTICE AUTH :*** Checking ident
NOTICE AUTH :*** No identd (auth) response
NOTICE AUTH :*** Found your hostname
```

Dann melden wir uns mit dem Nickname »irc-telnet« und dem Realname »Telnet Chatter« an:

```
NICK irc-telnet
PING :1437972627
PONG :1437972627
USER irc-telnet wird ignoriert :Telnet Chatter
:freenode.net 001 irc-telnet :Welcome...
(...)
:freenode.net 372 irc-telnet :- End Of MOTD
:freenode.net 376 irc-telnet :End of /MOTD.
```

Der »PONG«-Befehl als Antwort auf »PING« beeinflusst die Login-Prozedur nicht; Der Server prüft mit einem »PING«-Befehl nur, ob der Client noch online ist, oder, negativ ausgedrückt, ob die Verbindung tot ist. Auf ein »PING«-Paket muss man immer möglichst schnell mit einem identischen »PONG«-Paket antworten.

Die meisten Server setzen vor den Usernamen ein Tilde (»~«), wenn sie keine Ident-Antwort (auf Seite 101) erhalten.

Als nächstes gehen wir in den Channel »#infothek«:

```
JOIN #infothek
:irc-telnet!~irc-telne@13.9.ak.net JOIN :#infothek
:freenode.net 353 irc-telnet = #infothek :irc-telnet @iblech
:freenode.net 366 irc-telnet #infothek :End of /NAMES list.
```

Gehen wir die Antwort der Reihe nach durch.

```
:irc-telnet!~irc-telne@13.9.ak.net JOIN :#infothek
```

Diese Zeile liest sich so: „Der User mit dem Nick »irc-telnet« und dem Usernamen »˜irc-telne« des Hosts »13.9.ak.net« hat den Channel »#infothek« betreten.“

```
:freenode.net 353 irc-telnet = #infothek :irc-telnet @iblech
```

Hier haben wir eine Zeile mit einem Statuscode, »353«. Laut RFC ist dieser Statuscode mit »RPL_NAMREPLY« belegt, wodurch der Sinn klar wird: „Im Channel »#infothek« befinden sich die User mit den Nicks »irc-telnet« und »iblech«.“ Das »@«-Zeichen vor »iblech« signalisiert, dass iblech in diesem Channel „Operator“ ist. Damit erhält er höhere Privilegien.

```
:freenode.net 366 irc-telnet #infothek :End of /NAMES list.
```

Der Event »366«, »RPL_ENDOFNAMES« signalisiert das Ende der »NAMES«-Liste.

Auf jeden fall: Wir sind drin! Und können jetzt eine Nachricht an alle im Channel befindlichen Leute[52] schicken:

```
PRIVMSG #infothek :Hallo?
```

Der erste Parameter des »PRIVMSG«-Befehls bezeichnet den Empfänger der Nachricht. Der Empfänger kann entweder ein ganzer Channel (erkennbar an einem »#«, »+« oder »&« als erstes Zeichen des Names) oder eine einzelne Person sein. Sendet man an eine Person, so ist diese Nachricht für den Rest des Channels unsichtbar. Man kann auch eine Nachricht an mehrere Empfänger schicken, indem mann jedes Ziel einfach mit einem einzelnen »,« (Komma) trennt:

```
PRIVMSG +tuxforum,#perl,iblech :Me Be Leet
```

Wenn uns jemand anderes eine Antwort schickt, sieht das dann so aus:

```
:iblech!~iblech@13.9.ak.net PRIVMSG #infothek :Hi
```

Das Format der Rückgabe ist dabei

[52] Oder Bots. . .

```
:nick!username@ip-oder-hostname PRIVMSG ziel :Text
```

Dabei kann »ziel« wieder entweder ein Channel oder eine einzelne Person sein:

```
:iblech!~iblech@13.9.ak.net PRIVMSG irc-telnet :Geheim
```

Diese Zeile sagt uns, dass »iblech« uns eine private Nachricht mit dem Inhalt »Geheim« geschickt hat.

Um einen Channel wieder zu verlassen, benutzt man »PART«:

```
PART #infothek
:irc-telnet!~irc-telne@13.9.ak.net PART #infothek
```

Schließlich können wir die Verbindung beenden:

```
QUIT
ERROR :Closing Link: irc-telnet[~irc-telne@13.9.ak.net]
(I Quit)
iblech@thestars theguide $
```

11.3 Befehlsübersicht

Befehl	Wirkung
»NICK nickname«	Als »nickname« registrieren
»USER username egal egal :Realname«	Anmelden
»LIST«	Alle Channels anzeigen
»JOIN channel«	Einem Channel beitreten
»PART channel :Grund«	Aus einem Channel austreten
»TOPIC channel :Thema«	Das aktuelle Thema von »channel« anzeigen (ohne zweiten Parameter) oder setzen
»KICK channel nick«	„Kicked“ »nick« aus »channel«
»INVITE nick channel«	»nick« eine Einladung für »channel« ausstellen
»PRIVMSG empfänger :Text«	Nachricht an Empfänger schicken
»NOTICE empfänger :Text«	Nachricht an Empfänger schicken
»WHOIS nick«	Informationen über »nick« anzeigen
»WHO nick«	Informationen über »nick« anzeigen
»WHO channel«	Informationen über alle Mitglieder von »channel« anzeigen
»MODE nick_oder_channel modes«	Modes setzen oder löschen
»AWAY :Grund«	Mit (ohne) Argument aufgerufen sich als abwesend (anwesend) markieren
»QUIT :Grund«	Verbindung trennen

11.3.1 Nachrichtenversand

In der Tabelle sind zwei Befehle genannt, die eine Nachricht an einen oder mehrere Empfänger schicken: »PRIVMSG« und »NOTICE«.

Beide verhalten sich in den meisten Fällen identisch, ihr Unterschied kommt nur zum Tragen, wenn an einer „Unterhaltung“ ein Bot, also ein Client, der automatisch antwortet, beteiligt ist: Bot sollen nicht auf »NOTICE«s reagieren. Ansonsten könnten Endlosschleifen entstehen:

```
<Person> !seen Bot2
<Bot1> Person: Bot2 was last seen
<Bot1> Person: on #testchannel 5 minutes and 42 seconds
ago.
```

»Bot2« erkennt seinen Nicknamen in »Bot1«s Antwort und hält diese für einen Befehl. Da es aber (z.B.) den Befehl »Person« nicht gibt, reagiert er mit einer Fehlermeldung:

```
<Bot2> Bot1: No such command: "Person"
```

Daraufhin meint nun aber »Bot1«, er sei angesprochen worden:

```
<Bot1> Bot2: Command does not exist: "No"
```

Nun denkt wieder »Bot2«, dass er angesprochen wurde, usw.

Deswegen gibt es »NOTICE«. Bots sollen auf Anfragen **immer** mit einer »NOTICE« antworten. Andere Bots hingegen sollen eingehende »NOTICE«s immer ignorieren. Damit sieht das Gespräch dann so aus (»NOTICE«s werden in vielen Clients anders angezeigt als »PRIVMSG«s, im Beispiel: »-nick- text«):

```
<Person> !seen Bot2
-Bot1- Person: Bot2 was last seen
-Bot1- Person: on #testchannel 5 minutes and 42 seconds
ago.
```

»Bot2« fühlt sich nicht mehr angesprochen, die Endlosschleife wurde verhindert.

11.3.2 Benutzerinformationen

Es gibt zwei Befehle, um Informationen über Nutzer anzeigen zu lassen, »WHOIS« und »WHO«:

```
WHOIS nick
:server 311 own nick ~user host * :Real
:server 319 own nick :@#channel +#infothek +tuxforum
:server 312 own nick server :Servername
:server 317 own nick 5 :seconds idle
:server 318 own nick :End of WHOIS list.
WHO nick
:server 352 own * ~user host server nick umodes :hops Real
:server 315 own nick :End of WHO list.
```

Gehen wir die Antworten wieder Zeile für Zeile durch:

```
WHOIS nick
:server 311 own nick ~user host * :Real
```

Diese Zeile sagt uns (»own«), dass der Benutzer mit dem Nickname »nick« den Usernamen »~user« auf »host« hat und sein Realname »Real« ist.

```
:server 319 own nick :@#channel +#infothek +tuxforum
```

»nick« hat Operator-Rechte in »#channel«, Voice-Rechte auf »#infothek« und ist normaler Benutzer in »+tuxforum«.

```
:server 312 own nick server :Servername
```

»nick« ist auf »server« eingeloggt.

```
:server 317 own nick 5 :seconds idle
```

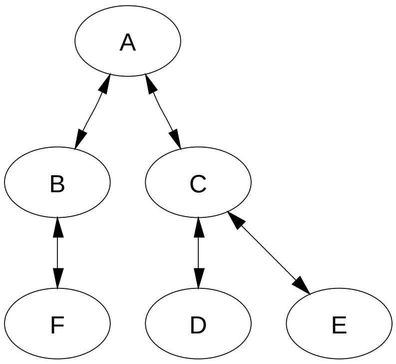
»nick« hat seit »5« Sekunden nichts mehr gesagt, hat also keine »PRIVMSG«s mehr verschickt.

```
:server 318 own nick :End of WHOIS list.
```

Diese Zeile markiert das Ende der »WHOIS«-Antworten.

```
WHO nick
:server 352 own * ~user host server nick umodes :hops Real
:server 315 own nick :End of WHO list.
```

»WHO« zeigt die Informationen kompakter an. Neu sind »umodes« und »hops«. »umodes« zeigt u.A. an, ob der Benutzer »AWAY« ist, mehr dazu im nächsten Abschnitt. Und »hops« ist eine Zahl, die angibt, wie viele „Hops" »nick«s Server von uns entfernt ist:



Wäre man selbst z.B. auf Server »B« und »nick« auf »E«, so würde »WHO« drei Hops anzeigen.

Mit Hilfe der Anzahl der Hops kann man auch das Lag, also die Zeit, die eine Nachricht von sich selbst zu »nick« benötigt, abschätzen – Befindet sich »nick« auf dem gleichen Server wie man selbst, so ist das Lag sicherlich relativ klein. Müssen Nachrichten allerdings erst mehrere Server passieren, bevor sie »nick« erreichen, ist das Lag größer.

11.4 Modes

Jedem Benutzer und jedem Channel sind bei IRC außerdem verschiedene Modes zugeordnet. Modes können gesetzt werden mit:

```
MODE user_oder_channel modes
```

Um z.B. sich selbst „unsichtbar" zu machen, kann man eingeben:

```
MODE mein_nick +i
:mein_nick MODE mein_nick :+i
```

Modes können auch ein Argument entgegennehmen. Um z.B. iblech im Channel #infothek Operator-Rechte zu verleihen kann man eingeben:

```
MODE #infothek +o iblech
:nick!user@host MODE #infothek +o iblech
```

Man kann auch mehrere Modes in einer einzigen Anweisung setzen:

```
MODE #infothek +oov-o iblech estel themaaaa themaaaa
:nick!user@host MODE #infothek +oov-o iblech estel themaaaa
themaaaa
```

Diese Anweisung würde den Benutzern iblech und estel in #infothek Operator-Rechte verleihen, themaaaa „voicen" und ihm anschließend seine Op-Rechte entziehen.

Je nach Server können einige Modes eine andere Bedeutung haben oder ganz fehlen, die meisten Modes sind jedoch in den IRC-RFCs definiert.

11.4.1 Benutzermodes

Benutzermodes, die man selbst, also per »MODE«-Befehl, setzen kann, sind:

Mode	Bedeutung
»+i«	„invisible": In globalen »WHOIS« oder »WHO«s wird man nicht aufgelistet.
»+w«	„see wallops": Man empfängt „Polizeifunk" – Opers können globale Nachrichten (»WALLOPS«) verschicken. Hat man »+w« gesetzt, empfängt man diese.

Es gibt aber auch noch die Umodes, die bei »WHO« angezeigt werden. Diese kann man nicht (direkt) selbst setzen.

Mode	Bedeutung
»H«	Man ist als anwesend markiert.
»G«	Man ist als »AWAY« markiert.
»*«	Man (oder besser: irgendjemand anderes) ist IRC-Operator.

11.4.2 Channelmodes

Mode	Bedeutung
»+b nick!user@host«	„ban“: Der angegebene Benutzer darf den Channel nicht mehr joinen. Der Parameter darf Wildcards enthalten. Ohne Argument zeigt der Server die Liste der gesetzten Bans an.
»+i«	„invite only“: Nur Benutzer, die »INVITE«d wurden, dürfen dem Channel beitreten
»+k schlüssel«	„key“: Setzt einen Channel-Schlüssel. Daraufhin ist ein »JOIN #channel« zwecklos, um zu joinen muss man den Schlüssel angeben: »JOIN #channel schlüssel«
»+l 42«	„limit“: Setzt die maximale Anzahl Benutzer, die den Channel joinen dürfen.
»+m«	„moderated“: Nur Channel-Operatoren und -„Voices“ dürfen reden.
»+n«	Man darf nur Nachrichten an den Channel senden, wenn man ihn auch betreten hat.
»+o nick«	Verleiht »nick« Operator-Rechte. Damit hat »nick« die Möglichkeit, andere Benutzer zu »KICK«en oder zu »INVITE«n und kann Channelmodes setzen.
»+s«	„secret“: Der Channel ist nicht in der Liste aller Channels zu sehen. Außerdem wird der Channel nicht in einem »WHOIS« auf seine Mitglieder angezeigt.
»+t«	Nur Ops dürfen das Channelthema ändern.
»+v nick«	„Voiced“ »nick«, siehe »+m«.

11.5 CTCP

Auf IRC aufbauend hat sich ein zweites Protokoll entwickelt, das Client-To-Client Protocol[53], CTCP.

Mit CTCP ist u.A. »/ME« realisiert – und »ME« ist **kein** IRC-Befehl. Der IRC-Client sendet bei der Eingabe von »/ME ist leet« folgendes an den Server (»^A« stellt ASCII-Byte »1« da):

```
PRIVMSG ziel :^AACTION ist leet^A
```

Die Syntax ist also: »^A«, CTCP-Kommando, Leerzeichen, Argument und abschließendes »^A«[54].

In vielen Client kann man mit »/CTCP ziel CTCP-Kommando Argument« manuell eine CTCP-Nachricht abschicken.

Einige CTCP-Befehle erwarten eine Antwort, diese ist dann als »NOTICE« zu senden:

```
PRIVMSG nick :^APING 1093955988^A
:nick!user@host NOTICE own :^APING 1093955988^A
```

[53] http://www.irchelp.org/irchelp/rfc/ctcpspec.html

[54] Laut CTCP-Spezifikation sind auch mehrere CTCP-Nachrichten und normale Nachrichten in einer einzigen IRC-Nachricht möglich, allerdings ist mir kein Client bekannt, der diese Möglichkeit auch umsetzt.

Befehl	Bedeutung
»ACTION ist leet«	Äquivalent zu »/ME ist leet«.
»PING argument«	Sobald ein Client ein CTCP-»PING«-Request bekommt, soll er ein »PING«-Response zurückschicken, **mit dem gleichen Argument**. Das Argument ist dabei oft ein Unix-Timestamp oder eine andere Zeitdarstellung. Sobald der Client die Antwort erhält, kann er dann einfach die aktuelle Zeit von der Zeit, die das Argument repräsentiert, subtrahieren und erhält das Lag zwischen sich selbst und dem Ziel.
»FINGER«	Liefert einige Benutzerinformationen wie Realname und Idle-Zeit zurück.
»VERSION«	Verschickt den Namen des Client, den das Ziel einsetzt, zurück.
»TIME«	Liefert die aktuelle Zeit in der Zeitzone des Ziels zurück.

11.6 DCC

Uns es gibt noch ein weiteres Protokoll, welches auf IRC und CTCP aufbaut: Direct Client Connection[55], DCC.

Will man Dateien direkt über IRC übertragen, so wäre die Geschwindigkeit sehr niedrig, da die meisten IRC-Server die Clients drosseln und sogar die Verbindung beenden, wenn zu viel Text in zu kurzer Zeit über die Leitung fließt („Excess Flood“).

Deswegen musste eine andere Möglichkeit her – DCC. Bei DCC werden die Nutzdaten direkt von Client zu Client übertragen, ohne Umweg über den IRC-Server. Nur das Initiieren der Verbindung wird über den Server, mittels CTCP, abgewickelt.

[55] `http://www.irchelp.org/irchelp/rfc/dccspec.html`

Die zwei am häufigsten genutzten Möglichkeiten von DCC sind der direkte Client-to-Client Chat und die Dateiübertragung.

Zum Aufbauen einer DCC-Verbindung schickt ein Client (A) einem anderen (B) folgende Nachricht:

```
PRIVMSG B :^ADCC typ argument host port größe^A
```

»größe« wird bei DCC-Chat nicht mitgesendet.

Um zum Beispiel die Datei »theguide.txt« an B zu senden, wird As Client nach Eingabe von »/DCC SEND B theguide.txt« folgendes senden:

```
PRIVMSG B :^ADCC SEND theguide.txt 167772163 32944 28424^A
```

Daraufhin wird B eine Verbindung zu »167772163:32944« herstellen. A hat diesen Port, »32944«, geöffnet und sendet dann die Datei.

Interessant ist die Kodierung des Host-Teils der Adresse, zu der B connecten soll: »167772163«

Schauen wir uns an, wie der Client zu dieser Adresse gekommen ist, die „echte“ Adresse ist »10.0.0.3«.

```
iblech@thestars theguide $ bc
bc 1.06
Copyright 1991-1994, 1997, 1998, 2000 Free Software Foundation,
Inc.
This is free software with ABSOLUTELY NO WARRANTY.
For details type 'warranty'.
obase=16
```

bc soll uns unsere Eingaben im Dezimalsystem ins Hexadezimalsystem konvertieren.

```
10
A
0
0
0
0
3
3
```

»10.0.0.3« hexadezimal aufgeschrieben ist also »A.0.0.3«.

Nun wollen wir »0A 00 00 03« als Hexadezimalzahl auffassen und sie zurück ins Dezimalsystem konvertieren:

```
obase=10
ibase=16
0A000003
167772163
(Strg+D um bc zu beenden)
iblech@thestars theguide $
```

Um wieder die IP-Adresse in „dotted-quad“-Form, also der „normalen“ Form, zu erhalten, könnten wir entweder die Schritte umgekehrt durchführen (Zahl ins Hexadezimalzahl konvertieren, in vier Teile von je zwei Ziffern zerlegen und diese dann wieder ins Dezimalsystem konvertieren) oder die Arbeit an ein anderes Programm übertragen, z.B. Ping:

```
iblech@thestars theguide $ ping -c1 167772163
PING 167772163 (10.0.0.3) 56(84) bytes of data.
64 bytes from 10.0.0.3: icmp_seq=1 ttl=64 time=0.039 ms

--- 167772163 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss
rtt min/avg/max/mdev = 0.039/0.039/0.039/0.000 ms
iblech@thestars theguide $
```

In der Tat können die meisten Programme, die eine IP-Adresse als Parameter erwarten, auch eine einzige große Dezimalzahl als IP-Adresse auffassen.

Man sollte niemals DCC-Angebote blind akzeptieren. Die mitgelieferte IP-Adresse muss nicht mit der IP-Adresse des Senders übereinstimmen. Das hätte zur Folge, dass man zu einem „wild fremden“ Computer connected. Ist zusätzlich der Port kleiner als 1024, so verbindet man sich sogar mit einem (unter Unix) privilegierten Dienst[56], den der „Sender“ willkürlich vorgibt. Je nach Dienst hat das andere Folgen.

Setzt man schlechte Betriebssysteme ein, so geht noch ein anderes Risiko von DCC aus, genau genommen nicht von DCC, sondern von den Dateien, die per DCC übertragen werden:

[56] Unter Unix können nur Programme mit Root-Rechten einen Socket an Ports unterhalb von 1024 binden.

```
<hax0r> N1mm mal eben den 1337 Download an, us0r
<us0r> Wieso, was ist das?
<hax0r> Nur nen kleines Spiel
<us0r> Aso, mom
(Nach Abschluss des Downloads)
<us0r> Hm, das Programm funktioniert nicht?!
* hax0r has quit (mwahahaha y00 got pwned)
<us0r> hax0r?
<us0r> Ehm
<us0r> Meine Maus spielt total verrückt
<us0r> Wargh irgendwas löscht meine Dateien!!
* us0r has quit (EOF from client)
```

Und ein weiterer infizierter Benutzer. . .

11.6.1 XDCC

Mit DCC kann man also einfach Dateien übertragen. Aber ein Problem gibt es noch, wenn man Dateien automatisch, ohne dass ein Benutzer eingreifen muss, zum Download anbieten will: Woher bekommen Clients die Liste der gespeicherten Dateien, und wie kann man einzelne Dateien abrufen?

Dazu gibt es XDCC. XDCC-Nachrichten sind, wie auch „normale" DCC-Nachrichten, in CTCP-Pakete verpackt:

Befehl	Wirkung
»XDCC LIST«	Der Server schickt dem Client (über IRC) die Namen der Dateien, die er anbietet.
»XDCC SEND #Paketnummer«	Der Server reiht den Clienten in eine Warteschlage ein. Sobald dem Server genug Bandbreite zur Verfügung steht sendet er dem Clienten einen DCC-SEND-Request, und die Datei wird übertragen.

12 DICT

Kommen wir nun zu etwas ganz anderem, der Online-Recherche:

Das Dictionary Server Protocol, definiert in RFC 2229[57], erlaubt den Zugriff auf sehr einfach zu bedienende, ASCII-basierte Online-Wörterbücher.

12.1 Design

Beim DICT-Protokoll gibt es üblicherweise mehrere Wörterbücher, die viele Begriffe aus unterschiedlichen Perspektiven definieren. Sucht man zum Beispiel im Wörterbuch »foldoc« nach „Windows“, so wird man eine objektive (=in diesem Fall schlechte) Antwort erhalten. Sucht man jedoch in der »jargon«-Datenbank, bekommt man die richtige Antwort zu sehen... ;-)

12.2 Beispielsitzung

Um etwas herum zu experimentieren, müssen wir natürlich erstmal eine Verbindung zu einem DICT-Server herstellen. Im Beispiel nehmen wir »dict.org« auf DICTs Standard-Port, TCP-Port »2628«:

```
iblech@thestars theguide $ telnet dict.org 2628
Trying 66.111.36.30...
Connected to dict.org.
Escape character is '^]'.
220 pan.alephnull.com dictd 1.8.0/rf on Linux 2.4.18-14
```

In der Bannerzeile ist neben den schon von SMTP oder NNTP bekannten Statuscodes die Versionsnummer des eingesetzten Serverprogramms enthalten.

Nun können wir eine einfache Suchanfrage starten:

```
DEFINE * "Munich"
150 2 definitions retrieved
151 "Munich" wn "WordNet (r) 2.0"
Munich
     n : the capital and largest city of Bavaria in
         southeastern Germany [syn: {Muenchen}]
.
151 "Munich" gazetteer "U.S. Gazetteer (1990)"
```

[57] http://www.faqs.org/rfcs/rfc2229.html

```
Munich, ND (city, FIPS 55020)
  Location: 48.66908 N, 98.83224 W
  Population (1990): 310 (135 housing units)
  Area: 1.6 sq km (land), 0.0 sq km (water)
.
250 ok [d/m/c = 2/0/120; 0.000r 0.000u 0.000s]
```

Wie bei SMTP wird jedes Ende der Ergebnisse (hier zwei, aus dem „U.S. Gazetteer (1990)“ und dem „WordNet (r) 2.0“) mit einem einzelnen ».« (Punkt) auf einer Zeile markiert.

Aber auch wenn man die exakte Schreibweise eines Begriffs nicht kennt, kann man das DICT-Protokoll benutzen:

```
SHOW STRATEGIES
111 8 databases present
exact "Match words exactly"
prefix "Match prefixes"
substring "Match substring occurring anywhere in word"
suffix "Match suffixes"
re "POSIX 1003.2 (modern) regular expressions"
regexp "Old (basic) regular expressions"
soundex "Match using SOUNDEX algorithm"
lev "Match words within Levenshtein distance one"
.
250 ok
MATCH * soundex "Hitchhiker"
152 55 matches found
web1913 "H toxicarius"
web1913 "Head-cheese"
web1913 "Headshake"
(...)
jargon "heatseeker"
foldoc "head disk assembly"
foldoc "heatseeker"
.
250 ok [d/m/c = 0/55/0; 0.000r 0.000u 0.000s]
```

Ist das gewünschte Wort gefunden, kann man wieder nach dem exakten Treffer suchen:

```
DEFINE jargon "heatseeker"
150 1 definitions retrieved
151 "heatseeker" jargon "Jargon File (4.3.0, 30 APR 2001)"
heatseeker n. [IBM] A customer who can be relied
   upon to buy, without fail, the latest
   version of an existing product (not quite
   the same as a member of the {lunatic
   fringe}). A 1993 example of a heatseeker was
   someone who, owning a 286 PC and Windows
   3.0, went out and bought Windows 3.1 (which
   offers no worthwhile benefits unless you
   have a 386).  If all customers were
   heatseekers, vast amounts of money could be
   made by just fixing some of the bugs in each
   release (n) and selling it to them as
   release (n+1). Microsoft in fact seems to
   have mastered this technique.


.
250 ok [d/m/c = 1/0/10; 0.000r 0.000u 0.000s]
```

Schließlich beenden wir die Verbindung mit »QUIT«:

```
QUIT
221 bye [d/m/c = 0/0/0; 216.000r 0.000u 0.000s]
Connection closed by foreign host.
iblech@thestars theguide $
```

12.3 Befehlsübersicht

Befehl	Wirkung
»DEFINE wörterbuch ”suchbegriff”«	Suche nach exaktem Treffer
»SHOW STRATEGIES«	Liste aller Suchstrategien
»MATCH wörterbuch strategie ”suchbegriff”«	Suche nach (unscharfem) Suchbegriff
»QUIT«	Sitzungsende

12.4 Anwendungen

Es gibt für gute Systeme verschiedene Programme, die DICT-Server abfragen können, z.B. kdict für das K Desktop Environment (im kdenetworks-Paket eingeschlossen). Auch kann man DICT, dank vorgefertigter Module für viele Programmiersprachen, auch sehr schön automatisieren (auf Seite 116).

13 Finger

Mit dem Finger-Protokoll, definiert in RFC 1288[58], ist es möglich, zu sehen, welche Benutzer auf einem Zielsystem eingeloggt sind. Obwohl heute das Protokoll wenn überhaupt nur noch auf alten Unix-basierten Computern unterstützt wird, hat es immer noch eine Daseinsberechtigung, vor allem in größeren LANs.

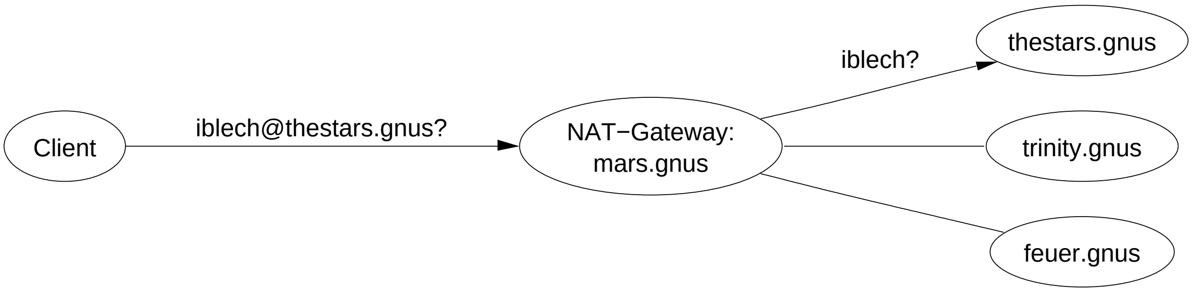
13.1 Design

Beim Finger-Protokoll werden, ähnlich wie bei E-Mails, Benutzer mit der Syntax »benutzername@host« eindeutig identifiziert.

Möchte man alle eingeloggten Benutzer aufgelistet haben, kann man auch den Benutzernamen weglassen, das heißt man fragt den Finger-Server nach »@host«.

Außerdem kann man »@host« weglassen, falls der User, für den man sich interessiert, auf dem Host ist, den man anfragt. »iblech« wäre also äquivalent zu »iblech@thestars.gnus«, wenn wir den Finger-Daemon von »thestars.gnus« abfragen. Ist der »host« im Request nicht der Host, den man gerade anfragt, so wird der Request (je nach Konfiguration des Finger-Daemons) an den „richtigen“ Host weitergeleitet, womit man also auch Benutzer innerhalb von ge-NATteten LANs „überprüfen“ kann:

[58] `http://www.faqs.org/rfcs/rfc1288.html`



13.2 Beispiel

Das Finger-Protokoll ist, ähnlich wie das Whois-Protokoll (auf Seite 112), ASCII-basiert und erwartet nur eine einzige Zeile als Eingabe, nämlich die „E-Mail-Adresse“ des Benutzers. Als Standard-Port kommt TCP-Port »79« zum Einsatz:

```
iblech@thestars iblech $ telnet thestars.gnus 79
Trying 10.0.0.3...
Connected to thestars.gnus.
Escape character is '^]'.
iblech

Welcome to Linux version 4.9.42-gentoo at thestars.gnus
!

 14:26:41 up 48 min,  7 users,  load average: 0.53, 0.29,
0.51

Login: iblech                          Name: Ingo Blechschmidt
Directory: /home/iblech                Shell: /bin/bash
On since Wed Sep  1 13:39 (CEST) on pts/2 from :pts/0:S.2
   10 minutes 32 seconds idle
On since Wed Sep  1 13:39 (CEST) on pts/3 from :pts/0:S.1
   33 minutes 55 seconds idle
On since Wed Sep  1 13:52 (CEST) on pts/4 from :pts/0:S.3
   23 seconds idle
On since Wed Sep  1 13:58 (CEST) on pts/5 from :pts/0:S.4
   10 minutes 2 seconds idle
On since Wed Sep  1 14:21 (CEST) on pts/6 from :pts/0:S.5
No mail.
No Plan.
Connection closed by foreign host.
iblech@thestars iblech $
```

Außerdem können mehrere Benutzernamen durch ein Leerzeichen voneinander getrennt und so mehrere Anfragen auf ein mal durchgeführt werden.

Ein besonderes Goodie ist die Suche nach Realnames: Je nach Konfiguration führt zum Beispiel eine Suche nach »Ingo@thestars« oder »Blechschmidt@thestars« auch zum Ziel!

13.3 Anwendungen

Auch heute noch nutzt die exzellente Adressdatenbank Little Brother´s Database[59] von Roland Roselfeld Finger:

> m_finger:
>
> This module will use finger to find out something more about a person. The list of hosts do be asked is configurable.

Mit ihrem Finger-Backend kann LBDB also u.A. den vollen Namen zu einer E-Mail-Adresse (Finger-Adressen sind ja oftmals identisch zu den zugehörigen Mail-Adressen) oder, nach Eingabe eines Teils des Reallife-Namens, die E-Mail-Adresse erhalten. Praktisch!

Außerdem ist Finger nützlich, wenn man wissen will, ob ein Nutzer gerade da ist oder nicht – Man fingert einfach seinen Account, und sieht sofort die Idle-Zeit.

14 Ident

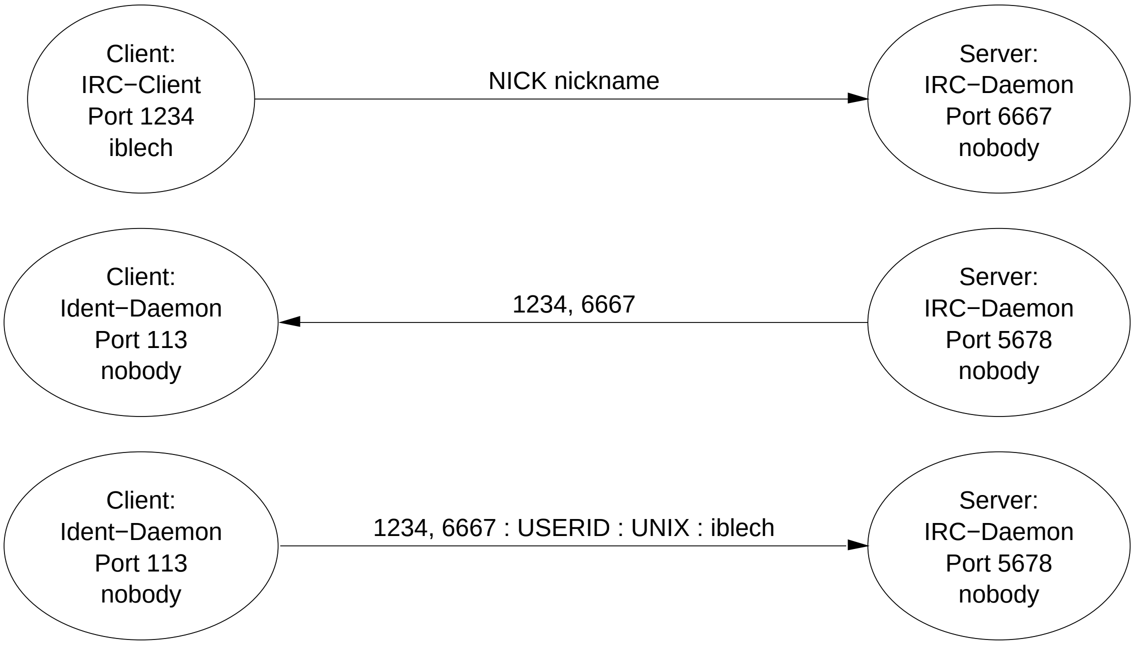
Ähnlich wie das Finger-Protokoll übermittelt das Ident-Protokoll (Identification-Protokoll), definiert in RFC 1413[60], keine Nutzdaten, sondern dient vielmehr zur Authentifizierung.

Über das Ident-Protokoll kann ein Server feststellen, wem (welchem Benutzer auf dem Client) die Verbindung „gehört“. So können zum Beispiel Verbindungen von root geblockt werden, zum Beispiel beim Chatten im IRC, weil das eine Sicherheitslücke darstellen würde – Würde der IRC-Client irgendwie komprommitiert werden, hätte ein Angreifer root-Rechte. . .

[59] http://www.spinnaker.de/lbdb/
[60] http://www.faqs.org/rfcs/rfc1413.html

14.1 Technische Realisierung

Um eine Verbindung eindeutig identifizieren, braucht man Quell- und Ziel-Port. Technisch umgesetzt sieht das dann so aus:



Testen wir das doch einmal. Dazu müssen wir zuerst auf TCP-Port »113«, dem Standard-Port von Ident, lauschen. Dann verbinden wir uns mit einem IRC-Server, z.B. »irc.gnu.org«. Daraufhin wird der Server bei uns anfragen, wem die Verbindung gehört:

```
mars root # nc -vlp 113
listening on [any] 113 ...
connect to [80.81.8.130] from leguin.acc.umu.se 33331
33443 , 6667
33443, 6667 : USERID : UNIX : iblech
mars root #
```

Wenn wir uns dann selbst »WHOIS«en, so wird als Username »iblech« angegeben sein, genau, wie wir gesagt hatten.

Der **zuerst** angegebene Port ist der Port **auf dem Rechner, auf dem der identd läuft**. Der zweite Port ist der Port auf dem Fremdrechner.

14.2 Anwendungen

Genutzt wird das Ident-Protokoll hauptsächlich bei IRC, wie man auch an den ersten Zeilen einer IRC-Verbindung feststellen kann:

```
iblech@thestars iblech $ telnet irc.gnu.org 6667
Trying 213.28.116.205...
Connected to irc.freenode.net.
Escape character is '^]'.
NOTICE AUTH :*** Looking up your hostname...
NOTICE AUTH :*** Checking ident
NOTICE AUTH :*** Found your hostname
NOTICE AUTH :*** Got ident response
```

Als ersten Teil der Authentifizierung führt der IRC-Server einen Reverse-Lookup (auf Seite 112) durch. Dann prüft er den Benutzernamen.

Wieso? Einige Würmer verwenden das IRC zum Nachrichtenaustausch. Einige Wurm-Programmierer sind aber zu faul, um auch einen Identd zu implementieren. So können IRC-Operatoren dann sehr schnell alle bösartigen Bots sperren – „Verbanne alle Benutzer, die keine Ident-Respone liefern und deren Fake-Username »xyz« enthält.“

15 Daytime

Als eines der wenigen „alten“ Protokolle abseits des Mainstreams wird das Daytime-Protokoll, definiert in RFC 867[61], auch heute noch sehr häufig benutzt.

Sobald eine Verbindung zu einem Daytime-Daemon hergestellt wird, übermittelt der Daemon das aktuelle Datum (mit Zeit) in menschenlesbarer Form und schließt daraufhin die Verbindung. So kann man z.B. die Systemuhren aller Rechner eines LANs abgleichen.

Wegen seiner geringen Komplexität kann Daytime sogar in Shellskripts eingesetzt werden (auf Seite 115). Laut RFC ist das Rückgabeformat zwar undefiniert, aber moderne GNU/Date-Implementationen sind in der Lage, viele verschiedene Formate richtig zu parsen.

[61] `http://www.faqs.org/rfcs/rfc867.html`

15.1 Beispiel

Nun wollen wir die aktuelle Zeit von »sombrero.cs.tu-berlin.de« abfragen. Dazu verbinden wir uns mit dem TCP- **oder** UDP-Port »13«:

```
iblech@thestars theguide $ nc -v sombrero.cs.tu-berlin.de
13
sombrero.cs.tu-berlin.de [130.149.17.8] 13 (daytime) open
Tue Aug 31 22:41:55 2004
iblech@thestars theguide $ nc -vu sombrero.cs.tu-berlin.de
13
sombrero.cs.tu-berlin.de [130.149.17.8] 13 (daytime) open
mdollar sux
Tue Aug 31 22:41:58 2004
(Strg+C)
iblech@thestars theguide $
```

Bei UDP gibt es ja, wie schon am Anfang erklärt, keine Verbindungen, deswegen müssen wir einfach irgendein Paket übermitteln, im Beispiel »mdollar sux«. Daraufhin schickt uns der Rechner der TU-Berlin die aktuelle Zeit. Abbruch wie üblich mit »Strg«+»C«.

16 DNS

Der Domain Name Service, DNS, ursprünglich definiert im 1983 eingereichten RFC 882[62], sorgt im Internet unter anderem dafür, dass man statt IP-Adressen (auf Seite 9) „symbolische" Namen eingeben kann, zum Beispiel »linide.sf.net«.

Zusätzlich ist es dafür verantwortlich, dass die E-Mail-Adresse »iblech@web.de« gültig ist, obwohl auf »web.de« kein SMTP-Server (auf Seite 24) lauscht. Mehr dazu weiter unten (auf Seite 111).

16.1 Geschichte

Am Anfang, als das Internet noch ARPAnet hieß und sehr viel kleiner war, wurden die IP-Adressen in einer sogenannten »hosts«-Datei übertragen, wie auch heute noch üblich in kleinen LANs. In

[62] http://www.faqs.org/rfcs/rfc882.html

so einer »hosts«-Datei[63] sind alle IP-Adressen mit ihren jeweiligen symbolischen Namen aufgelistet und mit Whitespace (alles, was auf dem Bildschirm leer erscheint, also zum Beispiel Leerzeichen und Tabulator) getrennt:

```
127.0.0.1 localhost      localhost.gnus
10.0.0.3  thestars.gnus thestars
10.0.0.4  mars.gnus      mars
```

Diese Liste wurde dann, immer wenn sie aktualisiert wurde, an **jeden** Teilnehmer des Internets übermittelt.

Als jetzt allerdings das Internet immer größer wurde und dementsprechend immer schneller wuchs, waren die Kosten für die Verbreitung der »hosts«-Datei nicht mehr zu tragen, weswegen ein neuer Dienst entwickelt wurde.

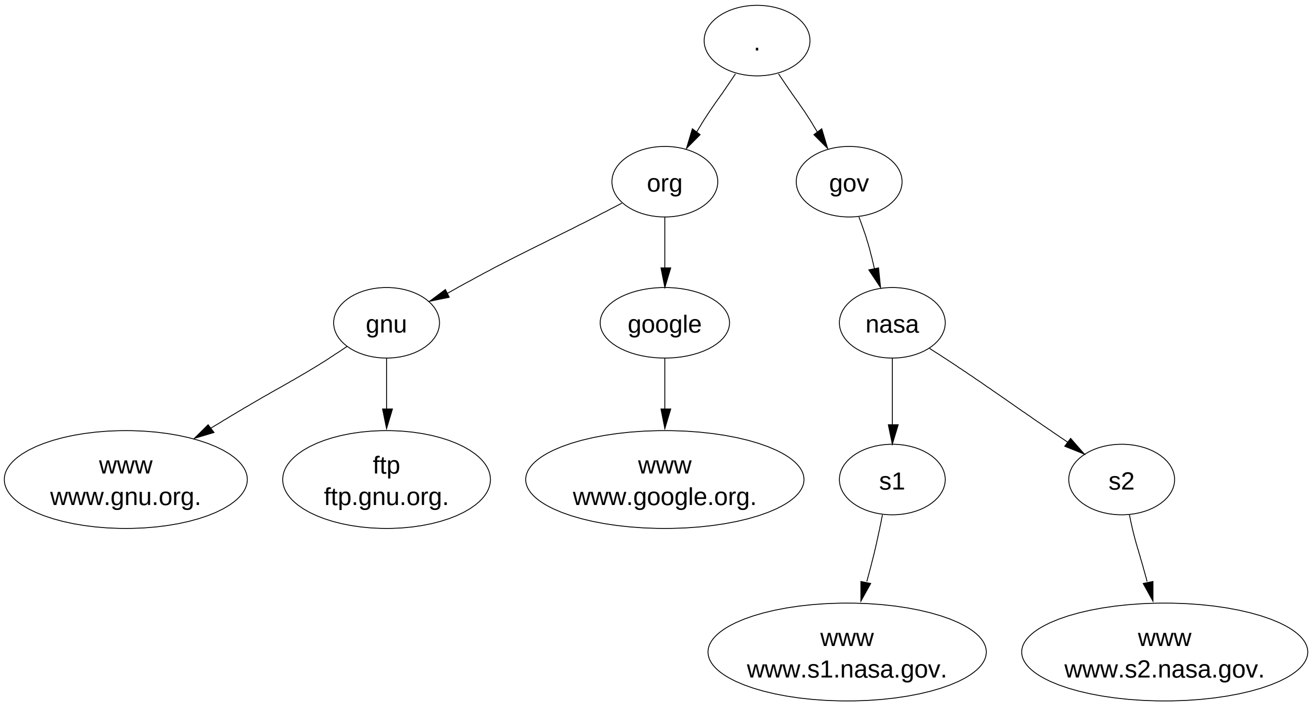
Man schlug einen zentralen Hostnames-Server, 1982 definiert in RFC 811[64], vor. Dieser sollte die Aufgabe haben, auf Anfrage eines symbolischen Namens die IP-Adresse des Rechners zurückzuliefern.

Zum Glück benutzen wir heute aber das über mehrere Server verteilte DNS-Protokoll.

16.2 Design

Beim DNS sind alle Rechner hierarchisch geordnet. Angefangen von der Wurzel, ».«, kann man Hostnamen beliebig weit verschachteln, jede Subdomain ist von ihrer Domain mit einem <<.>> getrennt:

[63] Auf guten Systemen findet sich die »hosts«-Datei in »/etc/hosts«.
[64] `http://www.faqs.org/rfcs/rfc811.html`



Wenn man nun zum Beispiel in einen Browser den Hostnamen »linide.sf.net.« eingibt, laufen einige Schritte ab, bis der Browser die IP des symbolischen Namens kennt:

- „Hey ».«, was ist die IP vom Nameserver von »net.«?“
- „Hey Nameserver von »net.«, was ist die IP vom Nameserver von »sf.net.«?“
- „Hey Nameserver von »sf.net.«, was ist die IP von »linide.sf.net.«?“

Erst wenn die endgültige IP-Adresse bekannt ist, kann der Browser via normalen HTTP die Seite anfordern. Da diese Schritte sehr oft wiederholt werden müssen, gibt es auch Caching-Nameserver. Da muss der Client nur den Caching-Nameserver nach z.B. »linide.sf.net.« befragen, und der Caching-Nameserver holt die Antwort **und speichert sie zwischen**. Kommt dann nochmal die gleiche Anfrage, so kennt der Nameserver die Antwort bereits und muss nicht nochmal ».«, »net.« und »sf.net.« befragen.

16.3 Vor- und Nachteile

Die Lösung des oben geschilderten Problems der Verteilung der hosts-Datei war damit natürlich gelöst: Jetzt brauchte nicht mehr jeder Rechner alle Zuordnungen der IPs zu symbolischen Namen kennen, da es ja jetzt die Möglichkeit gab, sich von der Wurzel ».« bis zum Ziel „durchzuhangeln“.

Heute gehört ».« der USA[65]. Wenn die USA nur eine einzige Zeile in der Konfigurationsdatei der Root-Nameserver (aus Gründen der Redundanz gibt es 26 Stück) ändern oder löschen würde, gäbe es kein »de.« mehr[66]. Das heißt, USA hat absolute Macht über das zentral-organisierte Internet[67].

16.4 Record-Typen

Nun zur technischen Seite von DNS. DNS ist vielmehr als nur ein hosts-Ersatz. Wo das Format der hosts-Datei nur je einer IP-Adresse einen oder mehrere symbolische Namen zuordnen kann, und keine hierarchische Ordnung besteht, unterstützt DNS viele verschiedene Record-Typen.

Aktuell gibt es u.A. folgende Record-Typen:

»A«
: Ein »A«-Record ordnet einem symbolischen Namen eine IP-Adresse zu.

»PTR«
: »PTR«-Records sind praktisch das Gegenteil der »A«-Records. Sie ordnen einer IP-Adressen einen (oder mehrere) symbolische Namen zu. Sie sind notwendig für Reverse-Lookups (auf Seite 112).

---

[65] no comment.

[66] Zumindest nach ein paar Stunden, wenn alle möglichen Caches aktualisiert sind.

[67] Allerdings kann jeder beliebiger Computer, wenn entsprechend eingerichtet, Root-Nameserver sein. So gibt es Bewegungen, wie etwa OpenNIC[68], die eigene Root-Nameserver anbieten und demokratisch organisiert sind. Dort gibt es auch andere Top-Level-Domains, zum Beispiel »geek.« und »parody.«.

»**NS**«
: Der »NS«-Record, der bei eigentlichen allen Dommains vorhanden ist, gibt die IP-Adresse des für das „Domain-Subnetz“ zuständigen Nameservers an. Dies ist notwendig, um das schon angesprochene „Durchhangeln“ zu ermöglichen.

»**CNAME**«
: »CNAME« ermöglicht es, einem symbolischen Namen einen anderen symbolischen Namen zuzuordnen. Dies nutzen zum Beispiel viele Hosting-Anbieter, die ihren Kunden eigene Domains ohne eigenen Server anbieten. In diesem Fall ist dann die Kundendomain nur ein Link auf DNS-Ebene auf einen der zentralen Server des Unternehmens.

»**MX**«
: »MX«-Records werden heute für den Mailverkehr gebraucht, wie gleich noch erläutert wird.

16.5 Ausfallsicherung

Wenn man sich ersthafte Gedanken über die Ausfallsicherheit eines bestimmten Servers macht, kommt man oft zum Schluss, ein sogenanntes Round-Robin-Verfahren anzuwenden.

Dabei konfiguriert man mehrere Server auf die gleiche Arbeit, zum Beispiel die Webpräsenz von IBM zu liefern. Dann benutzt man »A«-Records, um einem symbolischen Namen (zum Beispiel »www.ibm.com.«) mehrere IP-Adressen (die IP-Adressen der Server[69]) zuzuordnen.

Der Browser, bzw. die Name-Resolver-Bibliothek des Betriebssystems, pickt sich dann eine IP-Adresse heraus, die dann verwendet wird. So wird zum einen die Last gleichmäßig auf die verfügbaren Server verteilt und die Ausfallsicherheit erhöht (ist ein Server unerreichbar, wird der nächste genommen).

Ein schönes Beispiel dafür sind die rsync-Server von GNU Gentoo/Linux. Bei jedem erneuten Ping-Aufruf wird eine andere IP-Adresse angepingt:

---

[69] Oftmals benutzt man auch nur eine IP-Adresse und verwendet dann, ohne Hilfe von DNS, einen eigenen Server, der die Last selbstständig auf die anderen verteilt (Loadbalancer).

```
iblech@thestars theguide $ ping -c2 rsync.de.gentoo.org
PING rsync.de.gentoo.org (62.75.149.3) 56(84) bytes of data.
64 bytes from 62.75.149.3: icmp_seq=1 ttl=243
64 bytes from 62.75.149.3: icmp_seq=2 ttl=243

--- rsync.de.gentoo.org ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time
1003ms
rtt min/avg/max/mdev = 119.438/119.671/119.905/0.417 ms
iblech@thestars theguide $ ping -c2 rsync.de.gentoo.org
PING rsync.de.gentoo.org (213.131.230.230) 56(84) bytes
of data.
64 bytes from 213.131.230.230: icmp_seq=1 ttl=53
64 bytes from 213.131.230.230: icmp_seq=2 ttl=53

--- rsync.de.gentoo.org ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time
1000ms
rtt min/avg/max/mdev = 42.925/43.206/43.487/0.281 ms
iblech@thestars theguide $ ping -c2 rsync.de.gentoo.org
PING rsync.de.gentoo.org (141.12.220.13) 56(84) bytes of
data.
64 bytes from 141.12.220.13: icmp_seq=1 ttl=52
64 bytes from 141.12.220.13: icmp_seq=2 ttl=52

--- rsync.de.gentoo.org ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time
1011ms
rtt min/avg/max/mdev = 1339.558/1839.396/2339.235/499.840
ms, pipe 2
iblech@thestars theguide $
```

Unter Linux gibt es den Befehl »host«, der einige Informationen über symbolische Namen gibt. Ausgeführt mit »rsync.de.gentoo.org.« als Parameter erhält man:

```
iblech@thestars theguide $ host rsync.de.gentoo.org
rsync.de.gentoo.org has address 213.131.230.230
rsync.de.gentoo.org has address 212.224.22.34
rsync.de.gentoo.org has address 194.97.4.250
(...)
rsync.de.gentoo.org has address 80.190.246.242
rsync.de.gentoo.org has address 213.221.124.132
iblech@thestars theguide $
```

Noch mehr Informationen sind verfügbar mit der Option »-a«. Damit sieht man auch die zuständigen Nameserver:

```
iblech@thestars theguide $ host -a ibm.com
Trying "ibm.com"
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 47335
;; flags: qr rd ra; QUERY: 1, ANSWER: 20, AUTHORITY: 0

;; QUESTION SECTION:
;ibm.com.          IN   ANY

;; ANSWER SECTION:
ibm.com.   21558   IN   A    129.42.17.99
ibm.com.   21558   IN   A    129.42.16.99
ibm.com.   21558   IN   A    129.42.19.99
ibm.com.   21558   IN   A    129.42.18.99
ibm.com.   558     IN   NS   ns.austin.ibm.com.
ibm.com.   558     IN   NS   internet-server.zurich.ibm.com.
ibm.com.   558     IN   NS   ns.almaden.ibm.com.
ibm.com.   558     IN   NS   ns.watson.ibm.com.
ibm.com.   558     IN   SOA  ns.watson.ibm.com. \
  nrt.watson.ibm.com. 2004090700 3600 1800 604800 600
ibm.com.   558     IN   MX   10 e35.co.us.ibm.com.
ibm.com.   558     IN   MX   10 e34.co.us.ibm.com.
ibm.com.   558     IN   MX   10 e33.co.us.ibm.com.
ibm.com.   558     IN   MX   10 e32.co.us.ibm.com.
ibm.com.   558     IN   MX   10 e31.co.us.ibm.com.
ibm.com.   558     IN   MX   10 e6.ny.us.ibm.com.
ibm.com.   558     IN   MX   10 e5.ny.us.ibm.com.
ibm.com.   558     IN   MX   10 e4.ny.us.ibm.com.
ibm.com.   558     IN   MX   10 e3.ny.us.ibm.com.
ibm.com.   558     IN   MX   10 e2.ny.us.ibm.com.
ibm.com.   558     IN   MX   10 e1.ny.us.ibm.com.

;; ADDITIONAL SECTION:
ns.austin.ibm.com.             86358 IN A  192.35.232.34
internet-server.zurich.ibm.com. 21859 IN A  195.176.20.204

Received 501 bytes from 127.0.0.1#53 in 0 ms
iblech@thestars theguide $
```

16.6 MX-Records

Wie im Kapitel über SMTP (auf Seite 24) erläutert, muss es für jede Mail einen Ziel-SMTP-Server geben, der sich für sie zuständig fühlt. Dieser Servername wird eigentlich durch den Teil nach dem »@« der E-Mail-Adresse bestimmt. Aber wenn man mal einen einfachen Portscan (auf Seite 16) auf einen solchen Server (zum Beispiel »web.de«) durchführt, wird man feststellen, dass dort oft der SMTP-Service gar nicht angeboten wird. Wie kommt die Mail also trotzdem an?

Die Lösung dieses Rätsels liegt im »MX«-Record: Für praktisch jede Domain, die Mail erhalten soll, gibt es einen oder mehrere »MX«-Records. Diese Records stellen dann zum Beispiel die Zuordnung „der Mailserver für »web.de.« ist »smtp.web.de.«“ her.

So kann auch der Mailverkehr temporär auf einen Backup-Server umgeleitet werden, wenn der Haupt-Server zum Beispiel gerade gewartet wird.

Abfragbar[70] ist diese Information unter Linux mit Hilfe des schon erwähnten »host«-Befehls. Von Bedeutung sind diesmal aber nur die »MX«-Records, im Beispiel »mx-ha02.web.de« und »mx-ha01.web.de«. Auch hier gibt es also Redundanz: Fällt »mx-ha02.web.de« aus, so wird von den SMTP-Servern »mx-ha01.web.de« benutzt.

```
iblech@thestars theguide $ host -a web.de
Trying "web.de"
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 44459
;; flags: qr rd ra; QUERY: 1, ANSWER: 6, AUTHORITY: 0

;; QUESTION SECTION:
;web.de.                                IN      ANY

;; ANSWER SECTION:
web.de.           0       IN   A    217.72.195.42
web.de.           15354   IN   NS   nsx2.cinetic.de.
web.de.           15354   IN   NS   nsx1.cinetic.de.
web.de.           0       IN   SOA  nsx1.cinetic.de. \
  root.cinetic.de. 2004082602 28800 7200 604800 3600
web.de.           0       IN   MX   100 mx-ha01.web.de.
web.de.           0       IN   MX   110 mx-ha02.web.de.
```

[70]Diese Abfrage ist übrigens vollkommen legitim und legal. SMTP-Server führen exakt das gleiche aus, wenn sie Mails zustellen.

```
;; ADDITIONAL SECTION:
nsx2.cinetic.de.  70023   IN   A    217.72.194.213
nsx1.cinetic.de.  21252   IN   A    217.72.193.151
mx-ha01.web.de.   0       IN   A    217.72.192.149
mx-ha02.web.de.   0       IN   A    217.72.192.188

Received 245 bytes from 127.0.0.1#53 in 0 ms
iblech@thestars theguide $
```

16.7 Reverse-Lookups

Manchmal kennt mal aber schon die IP-Adresse und möchte den symbolischen Namen haben.

Um dies zu ermöglichen, könnte man entweder alle denkbaren Namen bruteforcen (schlecht) oder sich der »PTR«-Records bedienen (gut).

Nur für diese Reverse-Lookups wurde ein eigene Syntax entwickelt: Möchte man zum Beispiel den symbolischen Namen der IP-Adresse »10.0.0.3« erhalten, feuert man einfach einen Request nach »3.0.0.10.in-addr.arpa.« ab. Als Antwort erhält man dann den Namen (im Beispiel, bei mir lokal, »thestars.gnus.«).

Der Grund für das Umdrehen der IP-Adresse liegt in der Wertigkeit der einzelnen Abschnitte: Beim DNS ist der letzte Teil (die Top-Level-Domain) der höhstwertigste. Bei IP-Adressen ist es gerade umgekehrt: Die ».3« im Beispiel definiert nur die einzelne Node, die »10.« aber ein sehr großes Subnetz.

Übrigens nutzen viele SMTP-Server Reverse-Lookups aus, wenn sie überprüfen wollen, ob sie ihrem „Gesprächspartner“ trauen können: Sie gehen davon aus, dass, falls es möglich ist, die IP des Gegenübers in einen symbolischen Namen aufzulösen, der Remote-Server wohl einer Firma (= gut) angehören muss. Falls nicht, wird von einem Spammer ausgeganen und die Verbindung geschlossen.

16.8 Whois

Möchte man Informationen über Domains erhalten, benutzt man das Whois-Protokoll, welches von eigentlich allen Registrierungsstellen für Domains (NICs, Network Information Centers) angeboten wird.

Über Whois erfährt man zum Beispiel, wem die Domain gehört, wer der technische Verwalter ist und wer die Nameserver dieser Domain verwaltet.

Da Whois nur eine einzige Zeile als Eingabe erwartet, ist es sehr leicht zu bedienen: Möchte man zum Beispiel Informationen über die Domain »linux.de.« erhalten, so stellt man einfach eine Verbindung zum Whois-Server der NIC von »de.« (»whois.denic.de.«) auf TCP-Port »43« her und gibt ein:

```
linux.de.
```

ein. Die Antwort enthält dann neben der Copyright-Notiz und dem Disclaimer die gewünschten Informationen:

```
% Copyright (c)2002 by DENIC
%
% Restricted rights.
%
%
% Except for agreed Internet operational purposes, no
% part of this information may be reproduced, stored in
a
% retrieval system, or transmitted, in any form or by any
% means, electronic, mechanical, recording, or otherwise,
% without prior permission of the DENIC on behalf of
% itself and/or the copyright holders. Any use of this
% material to target advertising or similar activities
% are explicitly forbidden and will be prosecuted. The
% DENIC requests to be notified of any such activities
% or suspicions thereof.

domain:      linux.de
descr:       Christian Huettermann
descr:       Herrenberger Str. 14
descr:       D-72070 Tuebingen

nserver:     ns1.headlight.de
nserver:     ns2.headlight.de
status:      connect
changed:     20030514 194438
source:      DENIC
(...)
```

Das Ausgabeformat ist dabei abhängig von der NIC, deren Whois-Server üblicherweise »whois.nic.tld« (wobei »tld« für die Top-Level-Domain steht) zu finden ist.

Leider hat die DENIC WHOIS per WHOIS-Server vor kurzem **abgeschaltet**. Als ich diesen Abschnitt das erste Mal ende Juli 2003 verfasste, war eine Abfrage per Telnet auf »whois.denic.de«, Port »43«, noch problemlos möglich. Nun, September 2004, ist dies **nicht** mehr möglich. Stattdessen muss man das Webinterface[71] benutzen. Ich weiß nicht, was die DENIC eG dazu bewegt hat, WHOIS abzuschalten.

Bei anderen NICs funktioniert WHOIS aber zum Glück noch.

17 Muhahaha – oder: Automatisierung

In diesem Kapitel wird gezeigt, wie man die ganzen Protokolle, insbesondere HTTP, mit Linux-Tools automatisieren kann; Wer keine Linux-Kenntnisse hat, wird dieses Kapitel nicht interressieren.

17.1 Whois

Auch wenn man mit dem Whois-Protokoll sehr viele Informationen über Domains sammeln kann, ist das manuelle auffinden des zuständigen Whois-Servers auf Dauer mühsam, auch angesichts der Tatsache, dass es auch bei Whois Weiterleitungen gibt.

Deswegen gibt es für gute Systeme einige Whois-Clients, die diesen Vorgang automatisieren und auch den störenden Disclaimer automatisch entfernen.

Der Client »whois« ist dabei besonders praktisch, da er bei vielen modernen Distributionen gleich mitgeliefiert wird (herunterladbar von »http://www.linux.it/˜md/software/«). Die Bedienung ist überaus einfach, als Parameter wird einfach nur die Domain erwartet. Wichtig, wie bei manuellem Whois: Ein evtl. führendes »www.« muss natürlich weggelassen werden.

```
iblech@thestars theguide $ whois pro-linux.de
```

[71] http://www.denic.de/de/whois/index.jsp

```
% Copyright (c)2002 by DENIC

domain:      pro-linux.de
descr:       Mirko Lindner
descr:       Schafhaus 20
descr:       D-76698 Ubstadt-Weiher
descr:       Germany
nserver:     ns3.kundenserver.de
nserver:     ns4.kundenserver.de
status:      connect
changed:     20030616 151056
source:      DENIC

[admin-c]
Type:         PERSON
Name:         Mirko Lindner
(...)

[tech-c][zone-c]
Type:         PERSON
Name:         Gunther Stoecker
Address:      PCTOPIA Internet-Service
Address:      Lindenhof
(...)

iblech@thestars theguide $
```

17.2 Daytime

Das Daytime-Protokoll (auf Seite 103) ist sehr nützlich für die Synchronisation der Systemuhren vieler Rechner in einem Netzwerk. Aber wer will schon manuell in einem größeren LAN zu jedem Client gehen, »nc zeitserver 13« eingeben und dann möglichst schnell »date --set="[datim]"« ausführen müssen?

Da kann auch ein simples Shellskript eingreifen, was zum Beispiel als Cronjob[72] ausgeführt werden kann:

```
date --set="`nc zeitserver daytime`"
hwclock --systohc --utc
```

[72] Cronjobs werden vom crond (dem Cron-Daemon) regelmäßig zu bestimmten Zeiten ausgeführt. Sie können mit Hilfe des Kommandos »crontab -e« festgelegt werden.

Die letzte Zeile synchronisiert die Systemuhr von Linux mit der RTC-Uhr (Real Time Clock) des BIOS. Fehlt diese Zeile, so würde (zumindest auf einigen Distributionen) die Zeit nach einem Neustart wieder die alte sein.

Neuere Distributionen führen den »hwclock«-Befehl auch automatisch beim Herunterfahren aus.

17.3 DICT

Für Ruby gibt es ein Modul[73], mit dem man selbst Programme schreiben kann, die DICT-Server abfragen. Außerdem, und das ist auch für Menschen nützlich, die nicht Ruby beherrschen, liegt dem Paket ein Kommandozeilentool bei, »rdict«.

Mit »rdict« kann man bequem von der Shell aus DICT-Server abfragen:

```
iblech@thestars ruby-dict-0.9.2 $ ruby -Ilib/ ./rdict "linux
user group"
1 definition found

From The Free On-line Dictionary of
Computing (27 SEP 03) [foldoc]:

  Linux User Group

      <body, Linux> (LUG) Any
      organisation of {Linux} users in
      a local area, university, etc.,
      that offers mutual technical
      support, companionship with
      people of similar interests, and
      promotes the use of Linux among
      computer users generally.

      LUGs often hold Install Fests
      for the general public, in which
      experienced Linux users explain
      and supervise the installation
      of Linux on new users' systems.
```

[73] http://www.caliban.org/ruby/ruby-dict.shtml

```
          (2003-09-14)


    iblech@thestars ruby-dict-0.9.2 $
```

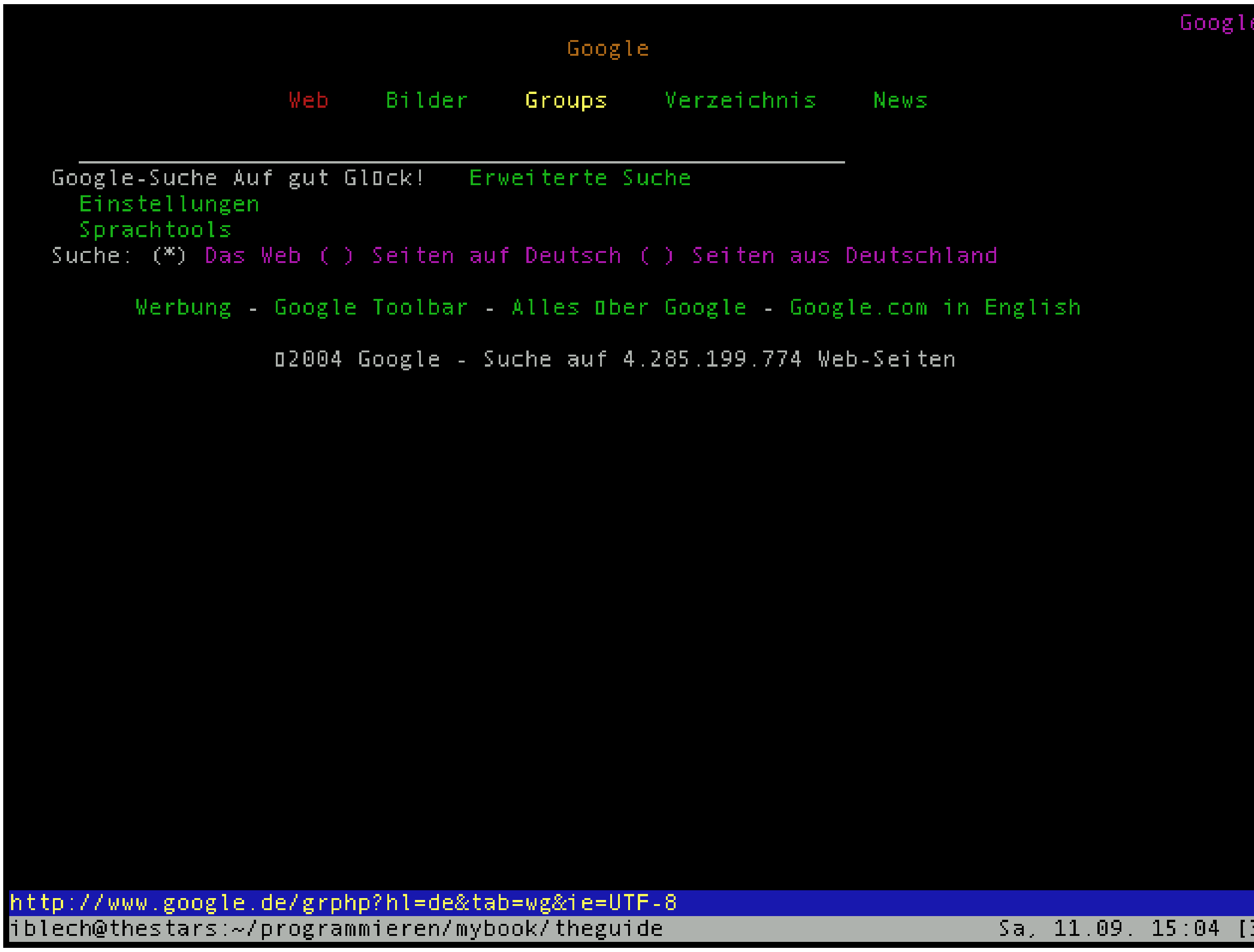
Einfach! :)

17.4 HTTP

Die Automatisierung bei HTTP ist natürlich besonders interessant, da heutzutage die meisten Dinge über HTTP laufen. Man kann zum Beispiel jeden Tag eine Website herunterladen und prüfen, ob sie seit dem letzten Tag geändert wurde, und, wenn ja, sie sich automatisch per Mail zuschicken lassen. Oder man manipuliert ein **bisschen** (grin) an Votes. . .

17.4.1 Lynx

Lynx, fast schon ein Klassiker, ist eigentlich nur zum text-basierten Surfen gedacht, lässt sich aber auch prima automatisieren.



Um zum Beispiel einfach den Sourcecode einer Seite abzurufen, benutzt man

```
iblech@thestars theguide $ lynx -source http://linide.sf.net/
```

Der Quelltext wird dann auf dem Terminal angezeigt bzw kann in Pipes weiterverarbeitet werden. Will man schon die gerenderte Version sehen, benutzt man

```
iblech@thestars theguide $ lynx -dump http://linide.sf.net/
```

Das Beispiel von oben, zu prüfen ob eine Website geändert wurde und gegebenenfalls per Mail verschicken, ist dann in Shell-Skript schnell implementiert:

```
URL="http://hackles.org/"
OMD5="$(md5sum < /tmp/seite)"
MAIL="iblech@web.de"

lynx -dump "$URL" > /tmp/seite
[ "$OMD5" = "$(md5sum < /tmp/seite)" ] || \
  mail -s "Aktualisierung von $URL" $MAIL < /tmp/seite
```

Ein GET-Request ist klar, wie im Kapitel über HTTP beschrieben (auf Seite 64), also einfach die Parameter mit einem »?« getrennt von der eigentlichen URL lynx als Parameter übergeben. POST-Requests sind ein bisschen schwieriger, sind aber auch sehr leicht möglich. Kommen wir zu unserem POST-Beispiel aus dem HTTP-Kapitel zurück: Wir wollen an »http://host/umfrage.p6« die Parameterpaare »name=Ingo«, »alter=16« und »os=Gentoo« per POST-Request senden. Mit Lynx sieht das dann so aus:

```
iblech@thestars theguide $ echo "name=ingo
> alter=15
> os=Gentoo
> ===" | lynx -post_data -source http://host/umfrage.p6
```

Daraufhin sendet Lynx die Parameter und gibt uns den Sourcecode der Ergebnisseite zurück.

Allerdings ist Lynx ein bisschen begrenzt, da man zum Beispiel nicht den Referer (auf Seite 56) setzen kann, den Lynx senden soll. Auch können Cookies nicht automatisch akzeptiert und in einer Datei gespeichert werden[74], usf.

17.4.2 curl

curl ist da schon wesentlich mächtiger. Neben der Fähigkeit, auch Dateien hochzuladen, die uns weniger interessiert, bietet curl volle Unterstützung für Cookies, GET- und POST-Requests, Location-Changing sowie den schon angesprochenen Referer.

[74] Was normalerweise unerwünscht ist, nämlich ständig Cookies anzunehmen, ist bei der Automatisierung sehr nützlich, manchmal sogar notwendig, da einige Seiten einen nur bei gesetzem Cookie akzeptieren

17.4.3 Standardsyntax

Um die eben erwähnten Features von curl zu aktivieren, sind einige Paramater nötig:

»-b /tmp/cookies$$«:
: Liest Cookies von »/tmp/cookies$$« ein. In jedem Skript sollte als eine der ersten Zeilen

```
> /tmp/cookies$$
```

stehen. Damit wird praktisch (am Bestern auch noch in Verbindung mit einem IP-Wechsel via Proxy) eine neue Identität angelegt.

»-c /tmp/cookies$$«:
: Schreibt neu erhaltene Cookies auf die Festplatte (bzw. in das Device, das auf »/tmp« gemountet wurde ;)).

»-e referer-url;auto«:
: Benutzt »referer-url« als Referer. Wenn das »;auto« angehängt wird, arbeitet curl exakt wie ein Browser (zum Beispiel wenn es einem »Location«-Header folgt).

»-L«:
: Erlaubt curl, »Location«-Headern zu folgen.

»-m 17«:
: Terminiert curl nach 17 Sekunden, egal was passiert. Ohne diese Option würde, wenn die Internetverbindung abreißt oder die Qualität der Proxies nicht gerade die beste ist, curl nie beenden und so das Skript aufhängen.

»-o ausgabedatei«:
: Setzt die Ausgabedatei.

Die Standardsyntax für curl lautet also

```
curl -b /tmp/cookies$$ -c /tmp/cookies$$ -e referer-url;auto
\
  -L -m 17 -o output url
```

GET-Requests

Wie schon bei lynx erwähnt, sind GET-Requests nichts anderes als verlängerte URLs, also besteht auch hier keine Notwendigkeit für eine neue Syntax. Es sei aber trotzdem angemerkt, dass curl mit der Option »-G« genau dies bietet.

POST-Requests

Ähnlich wie bei lynx, kann (muss aber nicht) auch bei curl ähnlich verfahren werden:

```
iblech@thestars theguide $ echo "name=Ingo&alter=15&os=Gentoo"
|
> curl -d @- (...)
```

Alternativ gibt es auch die Möglichkeit, von curls besserer Syntax gebrauch zu machen:

```
iblech@thestars theguide $ curl -d "name=Ingo" -d "alter=15"
-d "os=Gentoo" (...)
```

Mir gefällt allerdings folgendes Konstrukt am Besten:

```
{
  echo "name=Ingo"
  echo "alter=15"
  echo "os=Gentoo"
} | tr '\n' '&' | curl -d @- ...
```

So behält man selbst bei langen Feldern die Übersicht.

17.4.4 Perl-Modul HTTP::Recorder

Wenn man sich mit Perl auskennt, ist vielleicht auch das Modul HTTP::Recorder interessant.

Mit diesem Modul kann man sehr einfach eine Proxy schreiben, die alle Bewegungen (also auch Formularübertragungen) aufzeichnet und dann ein fertiges Skript produziert, welches nur noch ein bisschen angepasst werden muss.

Die Bedienung ist dank Proxy angenehm einfach: Einfach im Browser als Proxy das Perl-Programm setzen, welches HTTP::Recorder benutzt, und dann einmal den zu automatisierenden Vorgang manuell durchführen. Danach hat man das fertige Skript.

Das Modul kann man wie üblich mit der CPAN-Shell installieren:

```
iblech@thestars theguide $ perl -MCPAN -eshell

cpan shell -- CPAN exploration and modules installation
(v1.76)
ReadLine support enabled

cpan> force install HTTP::Recorder
(...)
```

Einen sehr guten englischen Artikel von Linda Julien ist auf perl.com[75] veröffentlicht.

[75] http://www.perl.com/pub/a/2004/06/04/recorder.html