In diesem Kapitel:
- *Netzwerke*
- *IP, TCP und UDP*
- *Das Internet*
- *Das Client Server Modell*
- *Internet Standards : RFC s*

2
Grundlegende Netzwerk Konzepte

2.1. Einführung

In diesem Kapitel wiederholen wir einige der grundlegenden Begriffe und Konzepte aus der Netzwerktheorie. Diese Kenntnisse benötigen Sie, um Netzwerkprogramme in Java oder einer anderen Programmiersprache schreiben zu können. Die Themen sind (top down) : generelle Netzwerkkonzepte, IP, UDP und TCP sowie das Internet. Die Konzepte werden nur skizziert, nicht von Grund auf erklärt. Hintergrundinformationen zu den Themen finden Sie beispielsweise im Buch von Tannenbaum oder den Büchern über TCP, speziell in der Ausgabe "illustrated": "TCP/IP Illustrated: Volume 1 The Protocols". Stevens, W. Richard. Addison Wesley Publishing Company, Reading MA, 1994. Wir gehen aber auch kurz auf den Standardisierungsprozess des Internets ein und besprechen Technologien wie Firewall und Proxy, dem Wesen nach.

2.2. Das Netzwerk

Ein Netzwerk ist eine Ansammlung von Rechnern und andern Geräten, die Daten senden und empfangen können, mehr oder weniger in Echtzeit. Ein Netzwerk wird normalerweise mittels Kabel verbunden und die Bits werden als Elektronen durch das Kabel gesandt. Daneben gibt es kabellose Verbindungen, bei denen die Informationen mit Infrarot oder Microwellen übertragen werden. Für höhere Geschwindigkeiten werden Lichtfasern eingesetzt. Aber welches Medium eingesetzt wird ist letztlich nur eine Frage der geforderten Gescgwindigkeit. Im Prinzip konnte die Information genauso mit Rauchsignalen übertragen werden, etwas langsam und mühsam, aber machbar, abgesehen von der Umweltverschmutzung und den schlechten Antwortzeiten.

Jedes Gerät im Netzwerk wird als *Node*, als Knoten bezeichnet. Die meisten Knoten sind Rechner, aber Drucker, Bridges, Gateways, Terminals und der Cola Automat können auch Knoten sein. Auch mit Java können Sie auf den Cola Automaten zugreifen. Ab 2000 ist das Thema "embedded systems" im Java Umfeld besonders aktuell. Viele Forscher und Entwickler bemühen sich, sogenannte Micro-Devices an die Netzwerke anzuschliessen, mit Hilfe von Java. Knoten, die voll funktionsfähige Rechner sind, werden auch als *Host* bezeichnet.

Jedes Netzwerkgerät besitzt eine *Adresse*, eine Serie von Bytes, mit deren Hilfe die Geräte eindeutig identifiziert werden. In der Regel stellt man sich unter der Adresse eine numerische Zahl vor. Allerdings handelt es sich einfach um eine Reihe Bytes, die in keiner Programmiersprache einem Datentyp (long, int, ...) entsprechen müssen. Je mehr Bytes in der Adresse vorkommen, desto mehr Geräte können adressiert werden. Je mehr Adressen möglich sind, desto mehr Geräte können also gleichzeitig an das Netzwerk angeschlossen werden.

Je nach Netzwerk erfolgt die Adressierung unterschiedlich. AppleTalk wählt die Adressen beim Start zufällig. Der Rechner prüft dann, ob die Adresse bereits vergeben ist. Wenn dies der Fall ist, wird eine neue Adresse vergeben und erneut getestet, bis schliesslich eine eindeutige Adresse gefunden wird.
Beim Ethernet ist die Vergabe weltweit geregelt. Die Hardware Hersteller erhalten Blöcke gültiger und eindeutiger Adressen zugeteilt. Die Ethernet Adresse ist somit fix mit der Hardware verbunden. Jeder Hersteller ist dafür verantwortlich, dass jede Adresse jeweils nur genau einmal eingesetzt wird, also vergeben wird.
Bei den Internet Adressen ist das Verfahren noch einmal anders. Eine zentrale Behörde, in den USA, vergibt die grossen Adressblöcke an Organisationen oder regionale Agenten. Diese wiederum teilen den grossen Anbietern von Internetadressen Adressblöcke zu. Der Endanwender erhält von seinem Provider eine oder mehrere Adressen zugeteilt. Will eine Firma einen grösseren Bereich von IP Adressen, muss ein spezieller Antrag an die Vergabestelle gestellt werden. Im Antrag muss die Netzwerkkonfiguration und deren Entwicklung über die nächsten Jahre beschrieben werden. Stimmt die Vergabestelle dem Antrag zu, wird der Provider, mit dem man den Anschluss ans Internet realisieren möchte, der Firma den beantragten Adressblock zuteilen. Der Vergabeprozess kann mehrere Monate dauern.

Andere Adressierschemata verwenden einfach lesbare Namen, letztlich basiert die Namensgebung in der Gesellschaft darauf, war aber früher unterschiedlich je nach Region. Allerdings können in Netzwerken die Namen oft geändert werden, ohne dass die Adresse sich ändert. Eine Adresse kann auch gleichzeitig unterschiedliche Namen haben.

Alle modernen Rechnernetzwerke unterstützen Paket-Switching. Dies besagt, dass die Daten im Netzwerk zu Paketen zusammen gefasst werden und dann als Pakete übermittelt werden. Jedes Paket, aber eben nicht jedes Byte, enthält Informationen zum Zielrechner und zum Quellrechner. Einer der Vorteile des Paket-Switching ist, dass dadurch die Auslastung der Verkabelung optimiert werden kann. Auf dem Kabel können sich gleichzeitig Pakete unterschiedlicher Kommunikationspaare befinden. Da jedes Paket eine klare Zieladresse besitzt, ist die Gefahr, dass falsche Pakete an eine Bestimmung gesendet werden, eher gering. Aber es ist möglich unterschiedliche Pakete zu analysieren und zu versuchen Meldungen als Ganzes zu rekonstruieren. Diese Art der Kommunikation unterscheidet sich wesentlich von der Telefonverbindung, bei der beide Telefonpartner eine Linie für sich haben, abgesehen vom Multiplexer kontrollierten Verbindungen.

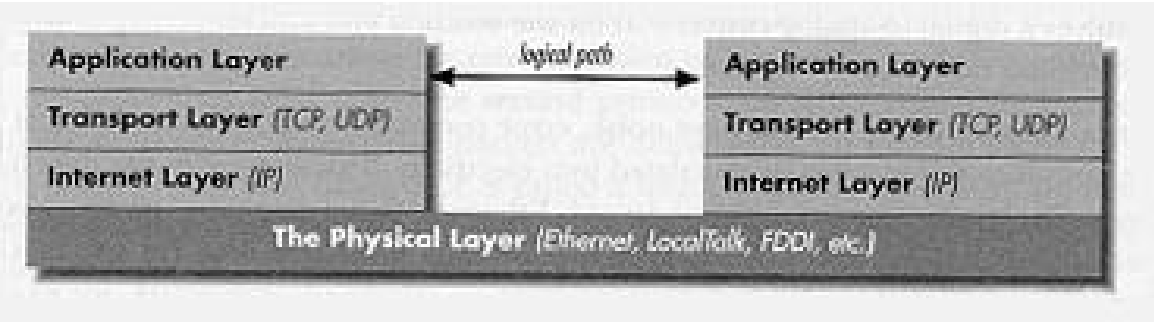
Ein wichtiges Element der Kommunikation fehlt bisher noch: das *Protokoll*. Das Protokoll ist ein Set von Regeln, mit deren Hilfe die Kommunikation zwischen zwei Kommunikationspartnern bestimmt wird. Im Protokoll wird festgehalten, wie die Rechner miteinander kommunizieren, wie die Adressen umgesetzt werden, wie die Daten in Pakete aufgeteilt werden, etc.
Es wurden sehr viele unterschiedliche Protokolle definiert und einige wurden auch implementiert. Beispielsweise das HTTP Protokoll, mit dem die Web Browser mit dem Server kommunizieren. In der Regel müssen Protokolle veröffentlicht werden, damit unterschiedliche Hersteller miteinander kommunizieren können. Die Zeiten der proprietären Protokolle sind also eher vorbei, wobei im Bereich der Router zum Beispiel CISCO teils mit eigenen Protokollen arbeitet, offiziell aus Effizienzgründen.

2.3. Die Ebenen eines Netzwerks (Layers)

In Netzwerken werden normalerweise unterschiedliche Ebenen, *Layer*, definiert. Jeder Layer definiert eine unterschiedliche Abstraktionsebene, von der physikalischen Ebene (Drähte,

Elektronen, Wellen) bis zur Informationsebene, zur Information die aus Sicht des Anwenders sinnvoll ist. Im Prinzip spricht jeder Layer jeweils nur mit dem direkt benachbarten Layer. Dadurch wird die Komplexität reduziert, das Kommunikationsproblem handhabbar.

Es gibt unterschiedliche Ebenenmodelle, jedes auf bestimmte Bedürfnisse ausgerichtet. Im Folgenden lehnen wir uns an ein 4-Ebenen-Modell an, eine Vereinfachung des klassischen ISO OSI Modells.



An Stelle von "Physical Layer" würde man besser vom "Host-to-Network" Layer sprechen.

Dieses Modell ist typischerweise dem Internet angepasst. Es beschreibt die wesentlichen Layer (IP, TCP, UDP und Applikationsebene) und lässt die tiefere Schicht (physical Layer) eher ausser Acht.

Der Hauptgrund für die Anwendung des Architekturpatterns "Layer" besteht darin, dass man versucht, die einzelnen Layer möglichst unabhängig zu machen und im Idealfall sogar austauschbar. Jeder Layer "spricht" lediglich mit seinem direkt benachbarten Layer. Der Applikationslayer spricht also lediglich mit dem Transportlayer; dieser spricht mit dem Internet-Layer (und dem Applikationslayer); der Internet-Layer seinerseits spricht mit dem Host-to-Network-Layer (physikalischen Layer) und natürlich dem Transport Layer.

Die echten Details der Protokolle sind jedoch, wie Sie sich vorstellen können, wesentlich komplexer. Die höchste Komplexität steckt im Host-to-Network Layer. Das soll uns aber nicht gross stören, da wir uns sehr häufig auf der Applikationsebene bewegen werden und uns kaum, noch kaum (das ändert sich mit der Java Micro Edition und der Einbindung der Microdevices), auf den unteren Ebenen bewegen.

Etwa 90% aller Applikationen die wir betrachten werden, benötigen lediglich Kenntnisse der oberen zwei Ebenen. Etwa 10% der Applikationen benötigt die zwei mittleren Layer. Unsere Beschränkung auf die vier Ebenen ist also durchaus berechtigt.

Dies zeigt sich auch an typischen Netzwerkapplikationen, wie etwa einem Chat Programm: der Chat Teilnehmer spricht mit einem Partner oder mehreren; in Wirklichkeit spricht er aber zu einem entfernten Rechner. Dieser benutzt selbstverständlich die Host-to-Network Verbindung; aber den Benutzer des Chat Programms kümmert dies kaum.

2.3.1. Der Host-to-Network Layer

Als Java Programmierer bewegt man sich in der Regel weit oben in der Kommunikationshierarchie. Das ändert sich aber laufend. Mit JavaCOMM steht ein Paket zur Verfügung, welches es Ihnen gestattet, die LPT's und COM Schnittstellen anzusprechen. Zudem werden mit neuen Profilen der Micro Edition von Java laufend neue Gebiete erschlossen (Connected, Limited Device Configuration (CLDC) der Java™ 2 Platform, Micro Edition (J2ME™).

Viel geschieht jedoch unterhalb des Applikationslayers. Im Standardmodell verschiebt man die Implementationsdetails in den Host-to-Network Layer. Dieser hier der unterste Layer besteht aus zwei Layern: dem physikalischen und dem Datalink Layer.

Der physikalische Layer besteht aus den Kabeln, den Glasfaserkabeln und andern Kommunikationsmitteln. Als Java Programmierer braucht man sich um diesen Layer in der Netzwerkprogrammierung nicht zu kümmern. Aber der CLDC Ansatz möchte diese Barriere durchbrechen.

Die eigentliche Kommunikation zwischen den Rechnern geschieht jedoch eigentlich nicht auf der physikalischen Ebene, sondern auf der Ebene der Applikationen. Dies umfasst im Wesentlichen die Interpretation der Datenpakete, die hin und her geschickt werden. Diese Interpretation ist im Protokoll geregelt.

Auf der untersten Ebene geht es dabei um Bits und Bytes. Diese Grössen sind digital, Ströme aber analog. Es muss also eine analog zu digital Umwandlung geschehen bzw. eine digital zu analog Umwandlung.

Da alle analoge Systeme mit Rauschen behaftet sind, müssen diese Fehler mit Hilfe von fehlerkorrigierendem Code oder andern Methoden behoben werden. Diese Aufgabe muss vom data link layer erledigt werden. Ethernet oder IEEE802.3 ist das am meisten verwendete Protokoll dieser Ebene. Andere Protokolle sind AppleTalk oder der Token Ring (in Zürich am IBM Forschungslabor entwickelt). Jedes Protokoll benötigt jeweils spezialisierte Hardware, also Spezialhardware für den Token Ring oder / und das Ethernet. Gateways zwischen den unterschiedlichen Protokollen werden häufig in Hubs und Switches integriert, als spezielle Karten. Als Java Programmierer kümmern Sie sich nicht um die Details auf dieser Ebene. Aber das kann sich ändern.

2.3.2. Der Internet Layer

Der nächste Layer des Netzwerks und der erste, der für den Java Netzwerkprogrammierer interessant wird, ist der Internet Layer. Im Internet Layer wird definiert wie Bits und Bytes zu Paketen zusammengefasst werden, also zu Paketen werden. Zusätzlich muss das Adressierschema beachtet werden. Das Internet Protokoll (IP) ist das am weitesten verbreitete Kommunikationsprotokoll. Java versteht zur Zeit lediglich dieses Layer Protokoll (beachten Sie die obigen Bemerkungen: COM und LPT werden mittels JavaCOMM, weitere Hardware Ebenen werden im CLDC angegangen). IPX von Novell ist ebenfalls sehr populär, wird von Java aber nicht direkt unterstützt. Auch NetBEUI von Microsoft wird nicht direkt unterstützt. Java orientiert sich also sehr stark an IP und TCP bzw. UDP, wie wir noch sehen werden. Es gibt aber verschiedene Implementationen der andern Protokolle in Java, die allerdings jeweils IP und TCP oder UDP als Träger verwenden.

Die Datenpakete des Internet Layers werden als *Datagramme* (engl. Datagram) bezeichnet. In einem IP Netzwerk besteht jedes IP Datagram aus einem Header mit zwanzig bis sechzig Bytes und Daten, bis zu 65'515 Datenbytes. Der Header jedes Paketes enthält eine Versionsnummer des benutzten Protokolls, verschiedene andere Nummern und vorallem die Adresse von dem das Paket stammt und die Adresse an die das Paket gesandt werden soll.

2.3.3. Der Transport Layer

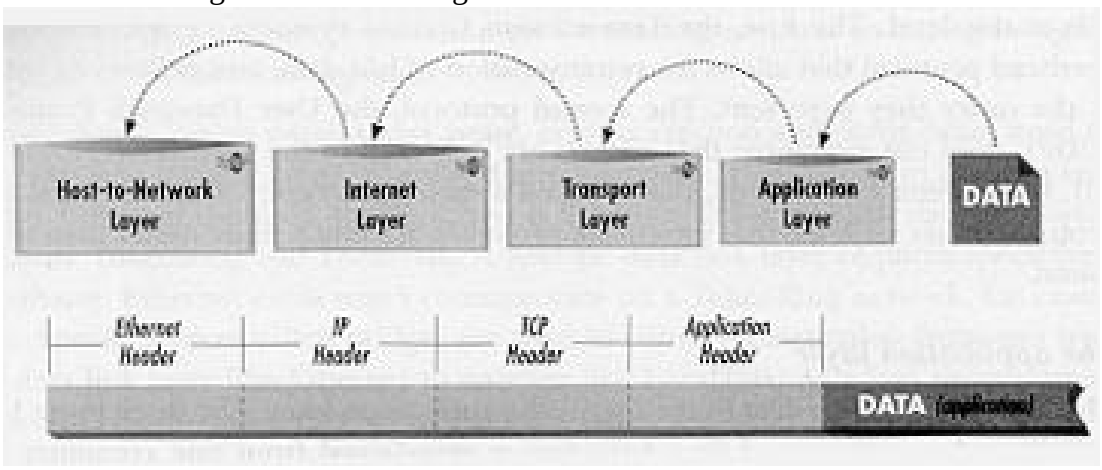
Reine Datagramme haben einige unschöne Eigenschaften: es ist nicht klar, ob und dass die Datagramme auch tatsächlich abgeliefert werden. Zudem ist unklar, in welcher Folge die

Pakete beim Empfänger ankommen. Dies ist die Aufgabe des Transport Layers. Dieser Layer muss dafür sorgen, dass die Pakete abgeliefert werden, dass dies in der richtigen Reihenfolge geschieht und dass keine Daten verloren gehen. Falls ein Datagramm verloren geht muss der Transport Layer den Sender bitten das Paket erneut zu senden. In IP Netzwerken wird dies mittels eines zusätzlichen Headers ermöglicht. Auf dieser Ebene gibt es zwei Protokolle. Das erste ist das Transport Control Protocol (TCP), welches mit einigem Aufwand garantiert, dass die Daten in der richtigen Reihenfolge abgeliefert werden. Das zweite Protokoll, UDP für User Datagram Protocol garantiert nicht, dass die Daten korrekt abgeliefert werden. Dafür werden die Daten schneller übermittelt, da eben viele Tests fehlen. TCP wird deshalb auch als zuverlässiges und UDP als unzuverlässiges Protokoll bezeichnet. Unzuverlässige Protokolle sind beispielsweise im Streamingbereich sehr wertvoll: es stört kaum, wenn von einem Musikstück einige Bits oder sogar Bytes fehlen; wichtig ist, dass die Daten schnell übermittelt werden.

2.3.4. Der Applikationslayer

Der Layer, welcher für die Ablieferung der Daten zuständig ist, ist der Applikationslayer. Die drei darunter liegenden Layer arbeiten zusammen, um die Daten korrekt beim entfernten Rechner abzuliefern. Der Applikationslayer entscheidet, was mit den Daten zu geschehen hat. Nehmen wir als Beispiel das HTTP Protokoll: die Daten, nehmen wir an es sei ein Bild, werden von den unteren Layern übertragen; aber nur der Applikationslayer kann das Bild darstellen. Analog verhält es sich bei anderen Protokollen wie SMTP, POP3, FTP oder was auch immer.

In diesem Zusammenhang ist der Begriff der *Kapselung*, encapsulation, wichtig: die Daten werden auf den einzelnen Layern jeweils in zusätzliche Informationen eingebettet, Header und eventuell Trailer Informationen. Die Header Information kann unter anderem Informationen enthalten, die dem Empfänger angeben, was er mit den Daten und wie er die Daten zu behandeln hat. Jeder Layer fügt seine spezifischen Informationen hinzu. Das Verfahren ist analog zum Einpacken eines Umschlags in einen grösseren Umschlag, und wieder in einen grösseren Umschlag...



2.4. IP, TCP und UDP

IP, das Internet Protokoll, besitzt eine Menge spezieller Vorteile im Vergleich mit anderen Protokollen wie AppleTalk oder IPX. Dies ist unter anderem historisch bedingt. IP wurd in der Zeit des kalten Krieges entwickelt, mit viel Militärgeld. Das Protokoll sollte vorallem robust sein und fehlertolerant implementiert sein. Die Fehlertoleranz wurde insbesondere mittels mehrfachem Routing implementiert: es sollte in der Regel möglich sein, dass Pakete

auf unterschiedliche Art und Weise, auf unterschiedlichen Routen, vom Sender zum Empfänger gelangen können. Damit können fehlerhafte Router umgangen werden.

Zudem war das Militär daran interessiert, die unterschiedlichsten Rechner an das Netzwerk anschliessen zu können. Das Protokoll musste also offen und plattformunabhängig sein. Die Rechner der damaligen Hauptlieferanten, IBM und DEC, sollten miteinander kommunizieren können.

Da die Pakete auf unterschiedlichen Wegen vom Sender zum Empfänger gelangen können, kann es passieren, dass je nach Netzwerkbelastung, jeweils ein anderer Weg der kürzeste ist, also vom Paket verwendet wird. Dies hat unter anderem zur Folge, dass die einzelnen Pakete nicht in der selben Reihenfolge beim Empfänger eintreffen. Um dieses Problem und einige andere Probleme zu beseitigen wurde das TCP Protokoll eingeführt. Mit Hilfe von TCP werden Sender und Empfänger in die Lage versetzt die einzelnen Pakete zu bestätigen, also den Empfang des Paketes beim Empfänger dem Sender mitzuteilen. Zudem werden mit TCP die einzelnen Pakete beim Empfänger wieder zusammengesetzt, in der selben Reihenfolge wie sie vom Sender generiert und gesandt wurden.

Allerdings führt TCP zu einem recht hohen Overhead. Falls es auf die einzelnen Pakete weniger ankommt und gelegentliche Verluste eines Paketes verkraftet werden können, bietet es sich an, ein anderes Protokoll, das UDP Protokoll zu verwenden. UDP Pakete werden nicht verifiziert (Reihenfolge, korrektes Übermitteln aller Pakete). UDP ist ein unzuverlässiges Protokoll, weil eben die Verifikation fehlt. Im Falle von Dateitransfer oder vielen anderen Applikationen ist die Reihenfolge und die korrekte Übermittlung aller Pakete wichtig. Beim Streaming-Video allerdings kaum, falls dadurch die Übertragungsgeschwindigkeit deutlich erhöht werden kann.

Als Java Programmierer braucht man sich nicht um die Details von IP zu kümmern, wohl aber um die Adressierung! Jeder Rechner besitzt eine eindeutige IP Adresse, zum Beispiel 62.88.20.22, wobei jede der Zahlen zwischen den Punkten einen Wert zwischen 0 und 255 annehmen kann. Wann immer Daten übermittelt werden, müssen die Daten mit einem Header versehen werden, in dem die Adresse (IP Adresse) des Zielrechners steht. Zusätzlich wird auch noch die Adresse des Senders notiert. Die Aufgabe des Routers ist es den optimalen Weg für die Datenpakete zu finden, aufgrund der Informationen im Header. Ohne Quelladresse wüsste der Empfänger nicht, wem er zu antworten hat.

Rechner können mit IP Adressen ganz nett umgehen. Für den Anwender ist dies eher unhandlich. Daher wurde das Domain Name System (DNS) eingeführt, mit dessen Hilfe die IP Adressen als Text verschlüsselt werden können. An Stelle von 190.123.221.124 kann man damit auch einen Namen wie *meinHost.Joller-Voss.CH* verwenden. Java stellt Ihnen verschiedene Klassen zur Verfügung mit deren Hilfe Sie die IP Adressen oder den Namen eines Knoten in einem IP Netzwerk bestimmen können.

2.4.1. Ports

Das Adressieren der Knoten würde eigentlich ausreichen, falls jeder Knoten nur genau eine Funktion hätte. Dies ist aber nicht der Fall. Beispielsweise kann ein Rechner problemlos gleichzeitig eingesetzt werden, um emails zu senden und Dateien mittels FTP zu transferieren. Daher müssen die einzelnen Anfragen unterschieden werden können. Dies geschieht mittels den *Ports*. Jeder Knoten mit einer IP Adresse kann bis zu 65'535 Ports besitzen. Diese Zahl ist eher akademisch. Ports existieren nicht physisch, wie etwa LPTs und COMs. Es sind also

keine "Kabel" mit den Ports verknüpft. Jeder Port entspricht einem bestimmten Dienst, beispielsweise FTP, SMTP ...

Beispielsweise entspricht HTTP als Service in der Regel dem Port 80. SMTP also das elektronische Mail entspricht dem Port 25. Wir werden Porttester schreiben, mit denen man nicht nur alle Ports erkennen kann, sondern auch (zum Teil) Testen kann.

Wann immer ein Paket an einen Host gesandt wird, wird es auch an einen bestimmten Port gesandt. Der Empfänger überprüft sowohl die Zieladresse als auch den Port. Anschliessend werden die Pakete an die Applikation gesandt, die für diesen Port zuständig ist, also Daten verarbeiten kann, die an diesen Port gesandt werden.

Port Nummern zwischen 1 und 1023 sind für vordefinierte Dienste reserviert. Sie finden unter Unix eine Datei mit einer Serviceliste, unter Windows ebenfalls (WINNT\System32\drivers\etc):

```
# Copyright (c) 1995 Microsoft Corp.
#
# Diese Datei enthält die Anschluánummern (Port Numbers) f• r bekannte
# Dienste gemäss RFC 1060 (Assigned Numbers).
# Bearbeiten Sie diese Datei mit einem ASCII-Editor.
#
# Format:
#
# <Dienstname> <Anschlussnummer>/<Protokoll> [Alias...] [#<Kommentar>]
#

echo            7/tcp
echo            7/udp
discard         9/tcp    sink null
discard         9/udp    sink null
systat         11/tcp
systat         11/tcp    users
daytime        13/tcp
daytime        13/udp
netstat        15/tcp
qotd           17/tcp    quote
qotd           17/udp    quote
chargen        19/tcp    ttytst source
chargen        19/udp    ttytst source
ftp-data       20/tcp
ftp            21/tcp
telnet         23/tcp
smtp           25/tcp    mail
time           37/tcp    timserver
time           37/udp    timserver
rlp            39/udp    resource      # resource location
name           42/tcp    nameserver
name           42/udp    nameserver
whois          43/tcp    nicname       # usually to sri-nic
domain         53/tcp    nameserver    # name-domain server
```

```
domain          53/udp    nameserver
nameserver      53/tcp    domain      # name-domain server
nameserver      53/udp    domain
mtp             57/tcp              # deprecated
bootp           67/udp              # boot program server
tftp            69/udp
rje             77/tcp    netrjs
finger          79/tcp
link            87/tcp    ttylink
supdup          95/tcp
hostnames       101/tcp   hostname    # usually from sri-nic
iso-tsap        102/tcp
dictionary      103/tcp   webster
x400            103/tcp             # ISO Mail
x400-snd        104/tcp
csnet-ns        105/tcp
pop             109/tcp   postoffice
pop2            109/tcp             # Post Office
pop3            110/tcp   postoffice
portmap         111/tcp
portmap         111/udp
sunrpc          111/tcp
sunrpc          111/udp
auth            113/tcp   authentication
sftp            115/tcp
path            117/tcp
uucp-path       117/tcp
nntp            119/tcp   usenet      # Network News Transfer
ntp             123/udp   ntpd ntp    # network time protocol (exp)
nbname          137/udp
nbdatagram      138/udp
nbsession       139/tcp
NeWS            144/tcp   news
sgmp            153/udp   sgmp
tcprepo         158/tcp   repository  # PCMAIL
snmp            161/udp   snmp
snmp-trap       162/udp   snmp
print-srv       170/tcp             # network PostScript
vmnet           175/tcp
load            315/udp
vmnet0          400/tcp
sytek           500/udp
biff            512/udp   comsat
exec            512/tcp
login           513/tcp
who             513/udp   whod
shell           514/tcp   cmd         # no passwords used
syslog          514/udp
printer         515/tcp   spooler     # line printer spooler
talk            517/udp
ntalk           518/udp
```

```
efs             520/tcp                 # for LucasFilm
route           520/udp    router routed
timed           525/udp    timeserver
tempo           526/tcp    newdate
courier         530/tcp    rpc
conference      531/tcp    chat
rvd-control     531/udp    MIT disk
netnews         532/tcp    readnews
netwall         533/udp                 # -for emergency broadcasts
uucp            540/tcp    uucpd        # uucp daemon
klogin          543/tcp                 # Kerberos authenticated rlogin
kshell          544/tcp    cmd          # and remote shell
new-rwho        550/udp    new-who      # experimental
remotefs        556/tcp    rfs_server rfs# Brunhoff remote filesystem
rmonitor        560/udp    rmonitord    # experimental
monitor         561/udp                 # experimental
garcon          600/tcp
maitrd          601/tcp
busboy          602/tcp
acctmaster      700/udp
acctslave       701/udp
acct            702/udp
acctlogin       703/udp
acctprinter     704/udp
elcsd           704/udp                 # errlog
acctinfo        705/udp
acctslave2      706/udp
acctdisk        707/udp
kerberos        750/tcp    kdc          # Kerberos authentication--tcp
kerberos        750/udp    kdc          # Kerberos authentication--udp
kerberos_master 751/tcp                 # Kerberos authentication
kerberos_master 751/udp                 # Kerberos authentication
passwd_server   752/udp                 # Kerberos passwd server
userreg_server  753/udp                 # Kerberos userreg server
krb_prop        754/tcp                 # Kerberos slave propagation
erlogin         888/tcp                 # Login and environment passing
kpop            1109/tcp                # Pop with Kerberos
phone           1167/udp
ingreslock      1524/tcp
maze            1666/udp
nfs             2049/udp                # sun nfs
knetd           2053/tcp                # Kerberos de-multiplexor
eklogin         2105/tcp                # Kerberos encrypted rlogin
rmt             5555/tcp   rmtd
mtb             5556/tcp   mtbd         # mtb backup
man             9535/tcp                # remote man server
w               9536/tcp
mantst          9537/tcp                # remote man server, testing
bnews           10000/tcp
rscs0           10000/udp
queue           10001/tcp
```

```
rscs1           10001/udp
poker           10002/tcp
rscs2           10002/udp
gateway         10003/tcp
rscs3           10003/udp
remp            10004/tcp
rscs4           10004/udp
rscs5           10005/udp
rscs6           10006/udp
rscs7           10007/udp
rscs8           10008/udp
rscs9           10009/udp
rscsa           10010/udp
rscsb           10011/udp
qmaster         10012/tcp
qmaster         10012/udp
```

2.5. Das Internet

Das *Internet* ist das grösste weltweite IP Netzwerk. Es umfasst jegliche Art von Rechnern und umfasst auch Server in der Antarktis. Jeder Knoten besitzt eine eindeutige IP Adresse und das Netz gehört allen und niemandem. Es handelt sich also lediglich um einen Verbund unterschiedlicher Rechner, die mittels eines Standardprotokolls kommunizieren.

Im Englischen unterscheidet man zwischen Internet, dem globalen IP basierten Netzwerk und internet, einem beliebigen, eventuell auf ein Office begrenztes IP basierte Netzwerk. Intranet ist ein anderes Schlagwort. Dabei handelt es sich um in der Regel IP basierte Netzwerke innerhalb eines Konzerns oder einer Firma.

Damit das Internet, das globale IP basierte Netzwerk, problemlos funktioniert, muss garantiert sein, dass die Adressen eindeutig vergeben werden da sonst Zugriffskonflikte entstehen könnten.

2.5.2. Internet Adressklassen

Um die Probleme bei der Vergabe der IP Adressen zu reduzieren wurde eine Organisation geschaffen, die im Moment gerade im Umbruch ist, nicht zuletzt wegen der Kommerzialisierung des Internets. Falls eine Firma oder eine Organisation ein IP basiertes Netzwerk aufsetzen möchte, wird der Firma ein IP Block zugeteilt, wie bereits weiter vorne beschrieben. In der Regel werden die Adressen in Klassen eingeteilt: Class A Adressen werden keine mehr vergeben, Class B de facto auch nicht mehr, Class C Adressen kann man allerdings noch erhalten. Es ist auch ein neues IP Adressierschema im Gange, seit einigen Jahren, da die IP Adressen langsam aber sicher ausgehen.

Bei einer Class B Adresse werden die ersten zwei Bytes vorgeschrieben, die andern zwei Bytes können frei verwendet werden.

Beispiel: Class B Adresse 167.1; erlaubte Knotenadressen : 167.1.0.1 bis 167.1.255.254

Class	Byte 0	Byte 1	Byte 2	Byte 3	Address Range
A	0 Network	Host			1.0.0.0 to 127.255.255.255
B	10	Network	Host		128.0.0.0 to 191.255.255.255
C	110	Network		Host	192.0.0.0 to 223.255.255.255
D	1110	Multicast Address			224.0.0.0 to 239.255.255.255
E	11110	Reserved for future expansion			240.0.0.0 to 247.255.255.255

Die Adressierung hat auch Auswirkungen auf die Zugriffsmöglichkeiten auf ein Netzwerk. Man könnte beispielsweise mit Hilfe eines einfachen Filters alle Datenpakete und damit alle Zugriffe mit IP Adressen, die ausserhalb des eigenen Bereichs liegen, ausfiltern. In

Wirklichkeit werden mittels Routern und Firewalls wesentlich komplexere Schutzmechanismen aufgebaut.

Verschiedene Adressen haben spezielle Funktionen:
die Adresse 127 ist nicht vergeben. Typischerweise wird sie für Loopback Tests, etwa als 127.0.0.1, verwendet. Der symbolische Namen, der dieser Adresse zugeordnet ist, ist *localhost*. Die Adresse 0.0.0.0 ist eine Standard Sourceadresse, sie kann aber nie als Zieladresse verwendet werden.

Was geschieht mit den Class A Adressen?

Als das Internet ursprünglich entworfen wurde, reichten die 126 Class A Adressen : jedes Subnetzwerk kann damit etwas über 16 Millionen unterschiedliche Netzwerkknoten adressieren. Aber es gibt kaum Organisationen, die ein so komplexes Netzwerk verwenden werden oder sollten. Daher machen Class A Netzwerke wenig Sinn. Es werden daher auch keine Class A Adressen mehr vergeben. Aber die Netzwerke des MIT und der Stanford University verfügen weiterhin über ein Class A Netzwerk.

Die Class D und Class E Adressen sind ebenfalls speziell: Class D Adressen werden für Multicasting Anwendungen verwendet (wir werden diese noch explizit kennen lernen); Class E Adressen sind für zukünftige Erweiterungen vorgesehen.

2.5.3. Firewalls

Um die cleveren Internet Benutzer etwas in ihrer Freiheit einzuschränken wurden relativ schnell spezielle Hardware und Software Systeme entwickelt, mit deren Hilfe der Zugriff auf ein Netzwerk kontrolliert und eingeschränkt werden kann: *Firewalls*. Um Firewalls sinnvoll einsetzen zu können werden die Zugriffe auf ein IP basiertes Netzwerk oft nur über eine einzige Verbindung zentralisiert. Diese Verbindung wird dann mittels Firewalls kontrolliert.

Die einfachste Form einer Firewall besteht aus einem Paketfilter: dieser prüft die einzelnen Pakete und lässt nur Pakete aus dem Netzwerk und in das Netzwerk, die bestimmten Kriterien entsprechen. Die Filterung basiert in der Regel auf den IP Bereichen und den Ports. So kann zum Beispiel der Zugriff von Host 98.123.34.12 auf unser Netzwerk explizit verboten werden. Allerdings wird ein Hacker dann sehr einfach seine IP Adresse verändern oder den Paketen eine falsche Adresse mitgeben.

Auch ausgehende Pakete, beispielsweise Telnet (Passwort und Benutzername werden im Klartext übermittelt), können gefiltert werden, um externen Interessenten nicht eine offene Türe anzubieten.

Die Konfiguration einer Firewall lernen Sie an anderer Stelle kennen. Es lohnt sich, diese Technik genauer anzusehen!

Bei der Netzwerk Programmierung mit Java werden Sie kaum Probleme mit Firewalls haben: Java bietet verschiedene Möglichkeiten, kontrolliert durch Firewalls zu tunneln. Aber dieses Tunneln kann auch verboten werden, Firewall seitig.

2.5.4. Proxy Servers

Proxy Server sind teilweise vergleichbar mit Firewalls. Die Firewall verhindert den direkten Zugriff auf andere Rechner; der Proxy Server ist ein Zwischending. Ein Client kann eine Webseite, auf die er nicht zugreifen kann, von einem Proxy verlangen. Der Proxy Server

verlangt dann diese Seite vom externen Server und liefert sie an den Client ab. Proxies kann man auch für FTP und andere Dienste einsetzen. Der Proxy Server erhöht die Sicherheit eines Netzwerks, weil durch die Zentralisierung der externen Zugriffe lediglich ein Server von aussen sichtbar ist. Externe Interessenten kennen somit die IP und sonstige Adressen des internen Clients nicht.

Firewalls arbeiten in der Regel auf Transport und / oder Internet Layer; Proxies dagegen arbeiten in der Regel auf Applikationsebene. Der Proxy hat also detaillierte Kenntnisse über die betreffenden Applikationen (FTP, HTTP). Der Proxy hat aber in der Regel auch die Fähigkeit Pakete auszufiltern, die eindeutig nicht zum entsprechenden Applikationstyp gehören.

Da der Proxy alle Zugriffe kanalisiert, besteht die Möglichkeit, ähnlich wie bei Firewalls, bestimmte Webseiten einfach zu sperren. Beispielsweise könnte eine Firma den Zugriff auf http://www.playboy.com oder http://www.vatican.vc zu verbieten. Dies geschieht im Proxy einfach dadurch, dass diese Web Adressen nicht aufgelöst werden, also keine Daten aus diesen Adressen zur Verfügung stehen. Einige Firmen benutzen Proxies, um das Surfverhalten ihrer Mitarbeiter oder Studenten zu überwachen. Beispielsweise kann mit Hilfe des Proxies verhindert werden, dass alle die Playmate des Monats herunterladen, oder das Wort zum Sonntag. Aber in der Regel zeugt ein solches Überwachen nicht gerade von professionellem Management und ist speziell in den USA recht kontrovers.

Mit Proxy Servern wird vorallem ein lokales Caching ermöglicht. Dateien, die von mehreren Benutzern benötigt werden, werden nicht mehrfach ab dem Website geladen, sondern nur einmal in den Proxy und ab dann jeweils lokal. Falls der Proxy Server die Datei nicht findet oder sicher sein möchte, eine aktuelle Version der Datei zu erhalten, dann greift er wieder auf die Originalseite, auf die Originaldatei zu. Die meisten Online Anbieter, wie etwa AOL oder T-Online verwenden grosse Proxy Server, um den Netzverkehr einigermassen im Griff halten zu können.

Das grösste Problem der Proxy Server ist, dass sie lediglich einige wenige Protokolle verstehen können. Typischerweise sind dies HTTP, FTP und eventuell SMTP. Aber heute noch vollständig. Oft bietet es sich an in Java ein Protokoll gleich selbst zu implementieren und mittels HTTP (dem Body Part) die Datenpakete zu transportieren. Jini ist eine der Technologien, die sich dies zu Nutze macht, aber auch RMI in einigen Spezialfällen.

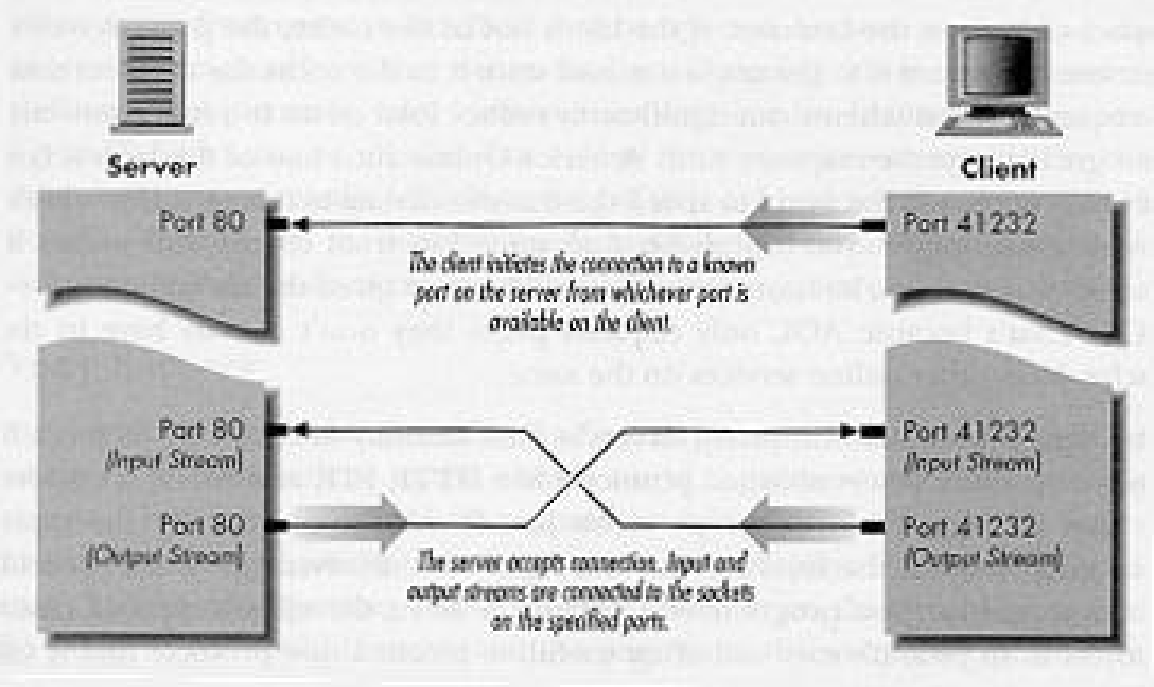
2.6. Das Client Server Modell

Client Server ist das Buzzword, welches viele während vieler Jahre entzückte. Auch heute noch versuchen sich viele ihre Applikationen aufzuteilen in einen Serverteil und einen Clientteil. In der Zwischenzeit ist man aber bereits bei Multitier Applikationen angelangt, welche mehr als zwei Teile (Client und Server), zum Beispiel Client, Applikationsserver und Datenbankserver, also drei Teile kennt. Clients sollen in der Regel aufwendige grafische Aufbereitungen abnehmen. Daten werden mit Vorteil auf einem leistungsfähigen Server gespeichert, in leistungsfähigen Datenbanken.

Der Unterschied zwischen einem Dateiserver oder Datenbankserver und einem Applikationsserver ist der, dass der Applikationsserver die Daten nicht nur speichert, sondern be- und verarbeitet, gemäss einer vordefinierten Businesslogik.

Die grösste Client Server Applikation ist das Internet: der Browser auf dem PC bereitet die auf Server abgespeicherten Daten grafisch auf. Auf der Serverseite sieht es heute so aus, dass

es viele sehr gute gratis Webserver gibt, wie zum Beispiel Apache. Auch kommerzielle Anbieter wie etwa IBM verwenden den Apache Server als Gerüst, um den herum ein eigentliches Produkt definiert wird. Sie sollten in der Lage sein, Apache und mindestens einen weiteren Webserver auzusetzen und zu konfigurieren.



Ablauf einer typischen Client Server Kommunikation:

- der Client initialisiert eine Verbindung zu einem Host / Port über einen Port des Clients
- der Server akzeptiert die Verbindung und definiert für die Kommunikation einen speziellen Port
- der Port 80 auf dem Server steht nun für weitere Anfragen zur Verfügung.

2.6.1. Peer-to-Peer Applikationen

Nicht alle Applikationen verwenden das Client Server oder das Multitier Modell. Einige verwenden ein "flacheres" Modell: das Peer-to-Peer Modell. In Online Spielen senden alle Spieler gleichberechtigt ihre Eingabe durch das Netz. Das System kennt also keinen dedizierten Server. In gewissem Sinne sind solche Systeme analog zu Telefonsystemen: Telefone kennen keine Server; die Kommunikation findet direkt zwischen den Partnern statt.

Java kennt kein spezielles API für diese Pier-to-Pier Kommunikation. Aber mit den java.net.* Klassen lassen sich leicht P2P Applikationen realisieren.

2.7. Internet Standards : RFCs

Im Folgenden werden Sie mehrere RFCs kennen lernen müssen und mit diesen Dokumenten arbeiten. Ein RFC = Request for Comments beschreibt beispielsweise ein oder Teile eines Applikationsprotokolls wie etwa SMTP, POP3, HTTP und viele andere. RFCs beschreiben Standards oder vorgeschlagene Standards sofern diese von der Internet Engineering Steering Group (IESG) der Internet Engineering Taskforce (IETF). Alle Internet Standards sind RFCs, aber nicht alle RFCs sind Internet Standards. Sie finden auf dem WWW mehrere Archive mit RFCs, beispielsweise bei SWITCH. Viele der RFCs sind technisch anspruchsvoll und nicht so einfach zu lesen. Aber oft handelt es sich um die einzigen verfügbaren Dokumentationen.

2.7.1. Wie werden Standards entwickelt

Typischerweise startet das Leben eines Standards bei einer Person oder einer Gruppe, die eine Idee hat und einen Prototypen baut. Der Prototyp ist wesentlich: bevor etwas als Internet Standard akzeptiert wird, muss es bereits implementiert sein. Dies garantiert, dass der mögliche Standard realisierbar, also keine Utopie ist. Falls der Prototyp populär wird, also andere Organisationen den Standard aufgreifen, wird eine Working Group innerhalb der IETF gegründet. Diese Working Group versucht das Protokoll als *Internet Draft* zusammenzufassen. Ein Internet Draft ist ein Arbeitspapier, mit dem eben gearbeitet wird: die Spezifikation kann sich nun noch gewaltig ändern. Nach mehreren Synchronisationen kann aus dem Draft ein Internet Standard werden. Falls dies der Fall ist, wird der Vorschlag an das IESG übergeben.

Danach bleibt der Status des Dokuments *experimental*. Dies hat aber keinerlei Einfluss auf die Verbreitung des Standards. Dieser kann bereits de facto akzeptiert sein. Viele der gängigen Standards sind immer noch Drafts.

Nach dem experimental Status wird ein Standard ein *proposed Standard*. Auch in diesem Stadium kann die Spezifikation noch jederzeit angepasst werden.

Nachdem die gröbsten Fehler eliminiert sind erreicht der Standard den Status *draft standard*. In diesem Stadium sollte sich die Spezifikation kaum mehr ändern.

Schliesslich erhält der Standard eine offizielle Nummer und wird damit zum offiziellen Standard. Bis hier sind mindestens 10 Monate ab erster Eingabe vergangen.

Neben diesem Lebenszyklus eines Standards gibt es auch noch einen *Requirement Level*:

- required
  alle Internet Hosts müssen diesen Standard implementieren. IP ist ein solcher Standard (RFC791). Andere Protokolle wie etwa TCP oder UDP sind nicht required sondern nur recommended.
- recommended
  diese Standards sollten von einem Host implementiert werden. Diese Standards umfassen die gängigen: TCP, UDP, FTP, SMTP, ...
- elective
  diese Standards können selektiv eingesetzt werden. Falls also ein Host ein bestimmtes Protokoll, wie etwa MIME (RFC1521) implementieren möchte, ist es ihm freigestellt dies zu tun. Die Implementation sollte dann aber auch RFC konform sein.
- limited use
  diese Standards sind nur für spezielle Anwendungen interessant
- not recommended
  diese Standards sollten nicht (mehr) implementiert werden
- historic
  sind veraltete Standards
- informational
  einige gute Konzepte, wie etwa NFS wurden nicht im Rahmen des RFC Prozess definiert, sind jedoch stark verbreitet. NFS ist im informational RFC1635 beschrieben