Wolfgang Hurst

The TCP/IP Guidebook

Version 0.5.2

Wolfgang Hurst:
The TCP/IP Guidebook - Version 0.5.2/ Wolfgang Hurst /

Textverarbeitungssystem : *vi*
Satz : LaTeX 2_ε Computer Modern 11 Pkt. (A4 Wide)
Graphische Tools : xfig, xv

INHALTSVERZEICHNIS

Kapitel 1

EINLEITUNG

Das TCP/IP[1] ist ein Netzwerkprotokoll. Heutzutage verbindet es die meisten Betriebssysteme miteinander. Der Vorläufer von TCP/IP, das ARPA[2] bzw. das ARPANET wurde vom amerikanischen Verteidigungsministerium DoD[3] ins leben gerufen. Zu den Zeiten des kalten Kriegs wollte das DoD ein stabiles und ausfallsicheres Verbindungsprotokoll für die zahlreichen Computer implementieren. Im Falle eines Atomangriffs sollten auf jeden Fall nur die beschaedigten Computer ausfallen und nicht das ganze Netzwerk. Vor dem ARPANET wurden die Verbindungen mittels serieller Leitungen direkt verbunden. Fiel eine Leitung aus stand meist der Rest des Netzwerkes still und musste umkonfiguriert werden. Aber nicht nur Atombomben sorgten fuer Ausfaelle, nein auch Unwetter jeglicher Art.

1.1 Dieses Dokument

soll nun einige Einblicke in das TCP/IP Protokoll geben. Aber nicht nur Einblicke sondern auch Konfiguration von TCP/IP auf verschiedenen Betriebssystemen. Aber auch die heutzutage unerlaessigen Subnetze und Supernetze, Multicasting und als leckerbissen das neue IPv6. Sowie die verschiedenen Standardprogramme und Protokolle. Mit kurzen Worten alles was mit TCP/IP zutun haben könnte

1.2 History

Dieser kleine Abschnitt hält die Entwicklung des Dokumentes fest. Die Zeichen haben folgende Bedeutungen :

n (new)	Thema hinzugefügt
d (deleted)	Thema entfernt
u (update)	Thema vervollständigt bzw. aktuallisiert
r (reorg)	Thema neu strukturiert
pn (part new)	Ein kompletten Teil hinzugefügt

Version 0.5.1 Released April 2000.

- n Diese History
- u ICMP - Domain Name Message (siehe 10.9 auf Seite 43)
- u NIS - Übersichtsgrafiken (siehe 45.9 auf Seite 196)

Version 0.4.7 Released Januar 2000. Diese Version war die erste die im CDI für die Allgemeinheit zur Verfügung gestellt wurde. Folgende Themen wurden geändert :

- pn NFS
- pn NIS

Version 0.3.8 Released August 1999

[1] TCP/IP-Transmission Control Protocol / Internet Protocol
[2] ARPA-Advanced Research Project Agency
[3] DoD-Departmanet of Defence

Version 0.0.1 Released June 1998

Kapitel 2

DoD - Schichtenmodell

Wie jedes vernümpftige Programm baut auch TCP/IP auf ein Schichtenmodell auf. Als Referenz wurde das hoffendlich bekannte ISO[1]-OSI[2] Modell rangezogen. Das ISO-OSI ist in 7 Schichten sehr fein aufgeteilt, das Modell das für TCP/IP entworfen wurde nennt sich DoD-Modell und besteht nur aus 4 Schichten. Die Grafik 2.1 veranschaulicht den Aufbau des DoD-Modells um das wir uns ersteinmal kümmern werden.

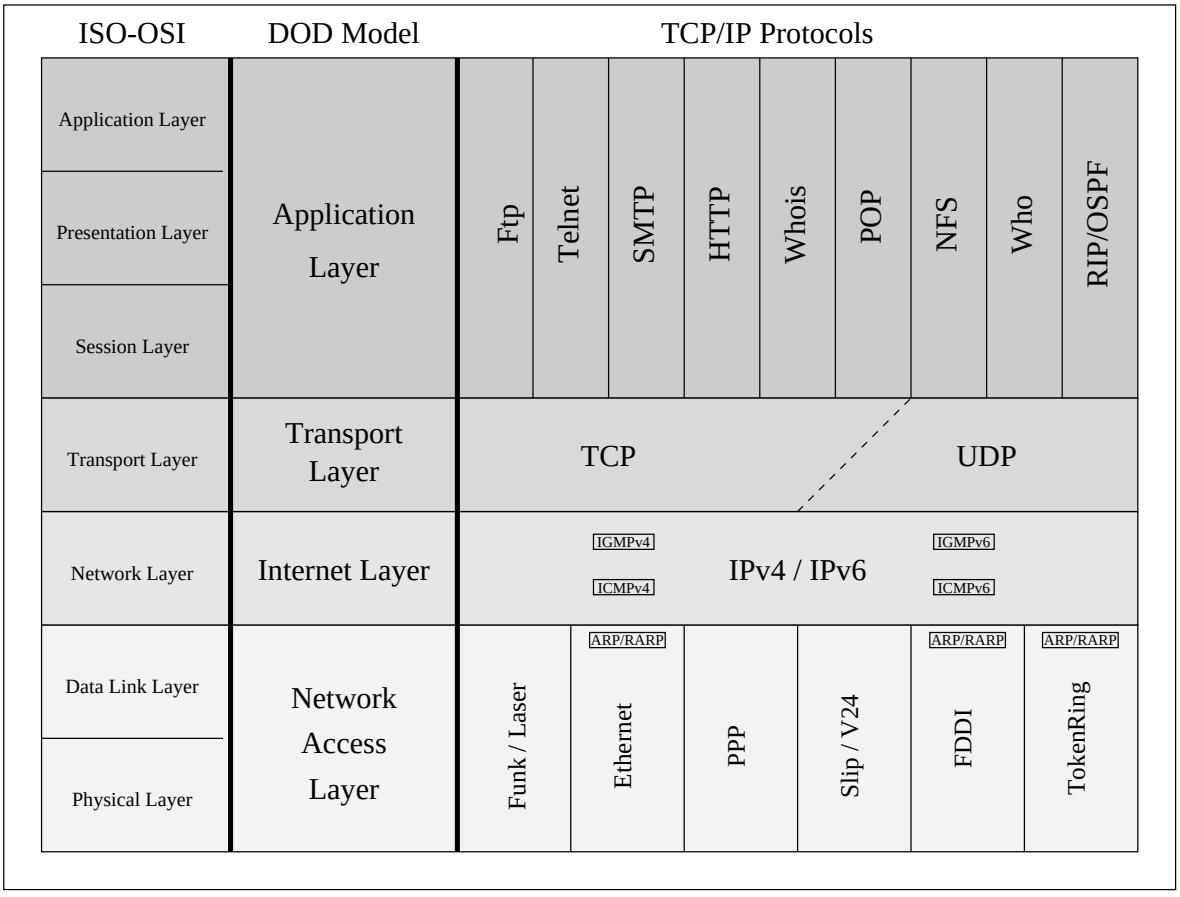

Abbildung 2.1: Das DoD Modell

Jede dieser einzelnen Schichten benutzt zur Erkennung einen Header der beim Senden zu den eigendlichen Daten hinzugefügt wird und beim Empfang wieder weggenommen wird. Somit wird eine 100%-ige Unabhängigkeit der einzelnen Schichten gewährt.

[1] `ISO`-International Standard Organisation
[2] `OSI`-Open System Interface

2.1 Network Access Layer

In dem NAL[3] wird definiert wie die Daten physikalisch transportiert werden sollen. Denkbar sind wie in der Grafik erkennbar die klassischen Ethernet Typen oder WAN[4] verwendete Protokolle wie PPP[5] oder Funk bzw. Laser Übertragungen. Der NAL bestimmt dabei die Art des Zugriffs sowie auch die verwendeten Komponenten. Beim Ethernet kommt das CSMA/CD[6] zum Einsatz, bei Laserübertragungen bestimmt der NAL die Art des Lasers, Wellenlänge etc. Bei SLIP[7] bestimmt hier der NAL den maximalen und den minimalen Pegel der Übertragung. SLIP ist jedoch vor einiger Zeit von PPP abgelöst worden, SLIP ist also nur noch selten zufinden, ältere Systeme jedoch kennen noch kein PPP. Der NAL bestimmt außerdem die MTU[8], da nur der NAL diese kennt. Der Ethernet_II Frame z.B. hat eine MTU von 1500 Bytes. Es entspricht also der größe der Nutzdaten.

2.2 Internet Layer

In dem IL[9] existieren zwei primäre Protokolle, IPv4[10] und IPv6[11]. Die beiden Protokolle beinhalten weitere kleinere Protokolle auf die wir später eingehen. Weiter werden wir die beiden Protokolle mit IP[12] umschreiben, und sofern nicht anderst angegeben gilt es immer für beide Protokolle. Der IL liegt auf dem NAL sinnbildlich drauf, es existieren diverse Schnittstellen damit der NAL mit dem IL, und umgekehrt kommunizieren kann. Der NAL ist aus der sicht des IL vollkommen unabhängig. Der IL funktioniert immer gleich, egal ob es nun übers Ethernet oder Modem geht. Aber mal zur eigendlichen Aufgabe des IL. Der IL stellt die virtuelle Verbindung zwischen Hosts zur Verfügung. Jeder Host bekommt eine eindeutige Kennung mit der er eindeutig identifiziert werden kann. Die IP-Addresse, zu der wir noch kommen.

2.3 Transport Layer

In dem TL[13] gibt es zwei Protokolle die sich um den Transport der Daten kümmern. Das UDP[14] hat im gegensatz zum TCP[15] die Eigenschaft ungesichert zu arbeiten. Nur mit TCP ist ein Datenaustausch gesichert. Unter UDP muß sich das Programm für die Sicherung kümmern.

[3] NAL-Network Access Layer
[4] WAN-Wide Area Network
[5] PPP-Point To Point Protocol
[6] CSMA/CD-Carrier Sense Multiply Access/Collision Detect
[7] SLIP-Serial Line Interface Protocol
[8] MTU-Maximum Transfer Unit
[9] IL-Internet Layer
[10] IPv4-Internet Protocol Version 4
[11] IPv6-Internet Protocol Version 6
[12] IP-Internet Protocol
[13] TL-Transport Layer
[14] UDP-User Datagram Protocol
[15] TCP-Transmission Control Protocol

TEIL I

NETWORK ACCESS LAYER

- Einführung
- Physikalische Übertragungen
  - Ethernet
- ARP
  - Funktionsweise
  - ARPv4
- Analyse und Konfigurationstools
  - ping
  - arp
  - tcpdump

Kapitel 3

EINFÜHRUNG

Der NAL beinhaltet viele Möglichkeiten um TCP/IP auf den verschiedensten Netzwerktopologien verfügbar zu machen. Da sich jedoch die Zugriffsart und die Addressierung der einzelnen Netzwerkkarten in den Topologien drastisch unterscheiden benötigt man eine Schnittstelle zwischen Hardwareaddressierung und Logischeraddressierung. Die Schnittstelle die man benötigt heißt ARP[1]. Das ARP ist bei FDDI etwas anderst aufgebaut als als unter dem herkömmlichen Ethernet, die funktionalität ist jedoch gleich. Daher wird es nur ein grosses Unterkapitel von ARP geben.

[1] ARP-Address Resolution Protocol

Kapitel 4

PHYSIKALISCHE ÜBERTRAGUNGEN

Physikalisch wäre wohl etwas zu heftig. Nein in dieser Section dreht es sich um Datenformate und technologien auf dem NAL.

4.1 Ethernet

In einem Ethernet, ganz gleich seiner physikalischen Eigenschaften, werden Daten zwischen einzelnen NIC[1]'s nur anhand der MAC[2] Addresse ausgetauscht werden. Jede NIC besitz eine auf der ganzen Welt eindeutige MAC. Eine MAC-Addresse besteht aus insgesamt 6 Bytes. Damit die Daten ausgetauscht werden können, unabhängig vom Protokoll, werden die zuübertragenden Daten in Frames verpackt. Die Frames sind standariesiert. TCP/IP benutzt in den meisten Fällen den einfachsten aller Frames, den *Ethernet_II* Frame. Die Grafik 4.1 veranschaulicht den Aufbau eines Ethernet_II Frames. Die Tabelle 4.1 schlüsselt die einzelnen Positionen auf.

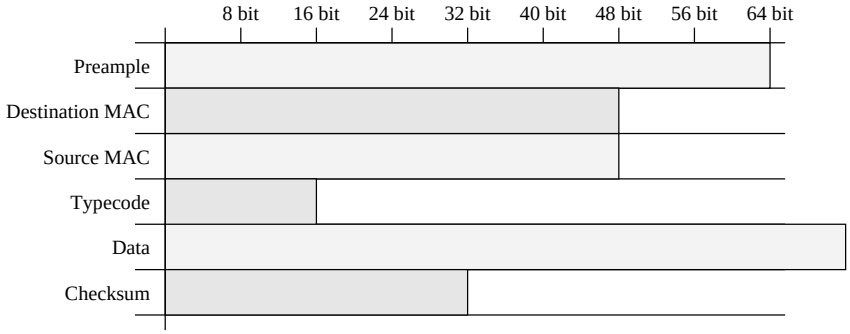

Abbildung 4.1: Ethernet_II Frametyp

Tabelle 4.1: Ethernet_II Frame Positionen

Feld	**Beschreibung**
Preample	Diese 8 Bytes werden zur Syncronisierung für die anderen NIC's gebraucht. 6 Bytes sind mit 55h Bitmustern gefüllt, die restlichen 2 Bytes werden gebraucht um den Frame starten zulassen, damit die anderen NIC's wiessen wann es endlich losgeht
Destination MAC	Die Ziel MAC-Addresse der NIC. Diese Zieladdresse ist vollkommen unabhängig anderer Addressen (wie IP). Man benoetigt ein Bindeglied zwischen MAC und logischen Addressen, unter TCP/IP übernimmt das das Protokoll ARP[3]

▼

[1] `NIC`-Network Interface Card
[2] `MAC`-Media Access Control
[3] `ARP`-Address Resolution Protocol

▲

Source MAC	Hier gilt das gleiche. Derjenige der diese Daten abgesendet hat, hinterlaesst hier seine MAC Addresse
Typecode	Der Typecode ist das entscheidene Feld in der Erkennung von Frametypen und Protokollen. Ist der Wert im Typecode kleiner als `0x05DC` (1500) handelt es sich nicht um einen Ethernet_II Frame sondern um einen Ethernet 802.[23] Frame, da in diesen Frames an dieser Stelle die Länge des Frames verzeichnet wird.
Nutzdaten	Die eigendlichen Daten die der nächsthöheren Schicht übermittelt werden
CRC[5]	Prüfsumme des Frames. Das CRC mit seinen möglichen Implementationen its in [RFC 1071][6]

Wert	**Protocol**
`0x0800`	IP
`0x0806`	ARP
`0x8035`	RARP[4]
`0x809B`	Apple Talk
`0x8137`	NetWare IPX/SPX

Es gibt noch andere Frames wie z.B. *Ethernet 802.2* etc. auf die wir jetzt noch nicht eingehen werden. Vielleicht irgendwann mal spaeter ...

[4] `RARP`-Reverse ARP
[5] `CRC`-Cyclical Redundancy Check
[6] `RFC-1071` Computing the Internet Checksum

Kapitel 5

ARP

Wie schon erwähnt benötigt man ARP um physikalische Addressen in logische Addressen umzuwandeln. Und weiterhin wissen wir bereits das es Unterschiedliche physikalische Addressierungen gibt. Aber auch unterschiedliche logische Addressierungen, genau genommen 2 einmal für IPv4 und IPv6, beide funktionieren jedoch genau gleich, nur mit den Unterschiedlichen IP Längen.

Solange das Kapitel für IPv6 noch nicht fertig ist werden wir uns nur um ARPv4 kümmern.

5.1 Funktionsweise

Kommen wir zur Funktionsweise von ARP. ARP wird gebraucht wenn eine logische Addresse vorhanden ist aber die Physikalische Addresse zur endgültigen Addressierung fehlt. Wenn z.B. ein Host addressiert werden soll ist meist nur die logische Addresse bekannt, also die IP, der NAL jedoch muß den Host mittels seiner physikalischen Addresse ansprechen, im Ethernet wäre es die MAC. Jeder Host führt daher eine TCP/IP interne Liste wo TCP/IP seine bekannten IP→MAC Addressen speichert. Diese Liste nennt sich ARP-Cache. Diese wird je nach bedarf aktualiesiert, bzw. wenn ein Eintrag längere Zeich nicht gebraucht wird wird dieser entfernt. Befindet sich jedoch die gewünschte logische Addresse nicht im ARP-Cache wird ein Broadcast gestartet, den Broadcast nennt sich ARP-Request. Der Host der seine logische Addresse im ARP-Request findet sendet ein ARP-Reply an den Host zurück. Danach weiß nun der anfragende Host die physikalische Addresse des Zielhost. Er speichert diese im ARP-Cache und kann nun endlich das Paket ausliefern. Aber ein Bild sagt mehr als 1024 Worte, die Grafik 5.1 auf der nächsten Seite zeigt den Ablauf.

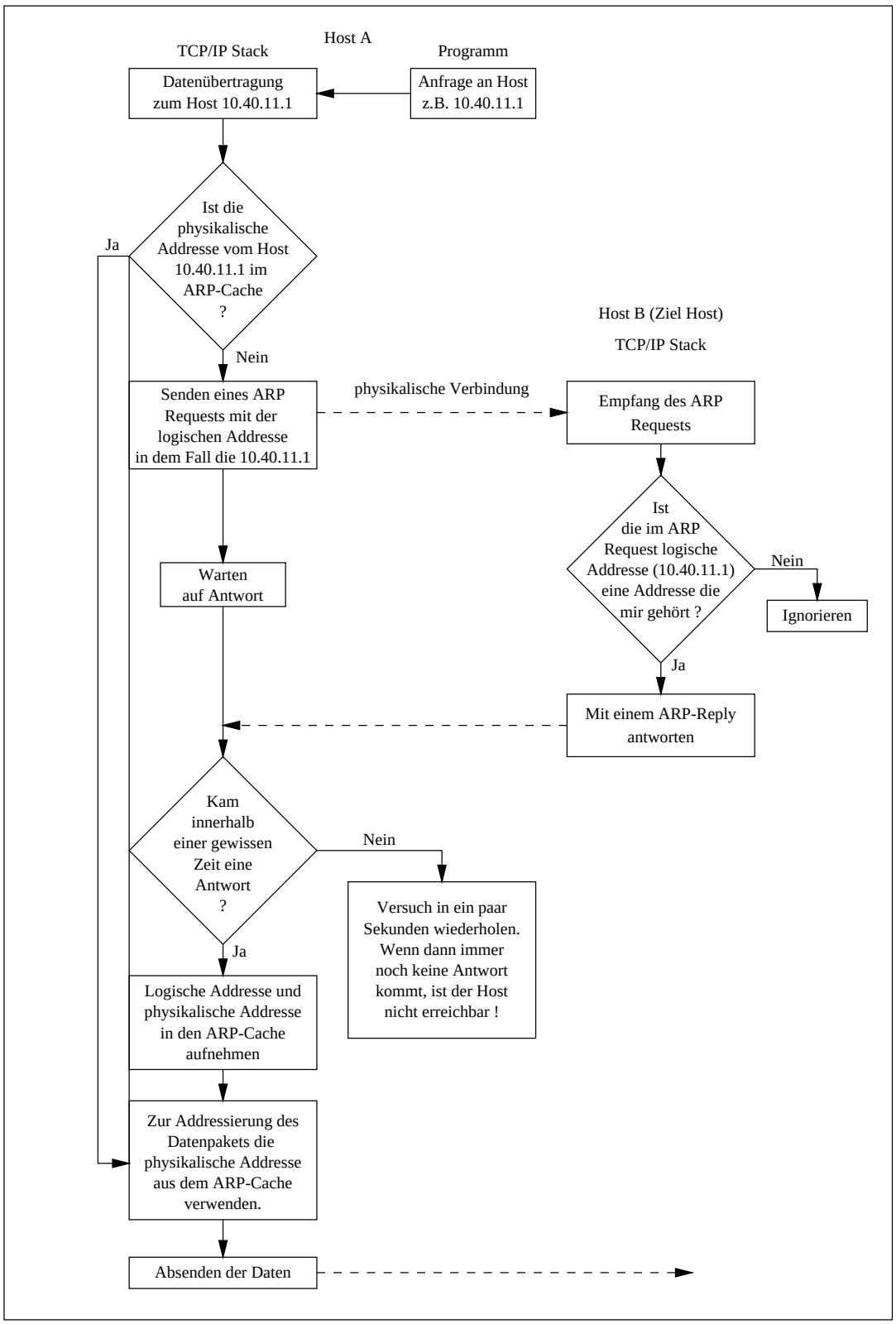

Abbildung 5.1: Funktionsweise von ARP

5.2 ARPv4

ARP-Request und ARP-Reply verwenden ein un den gleichen Header der direkt nach dem Frame angehängt wird. Der Aufbau des Headers zeigt die Grafik 5.2. Die einzelnen Felder dieses Headers werden in der Tabelle 5.1 aufgeschlüsselt.

Die Felder die bei einem Request unbekannt sind werden mit Nullen gefüllt.

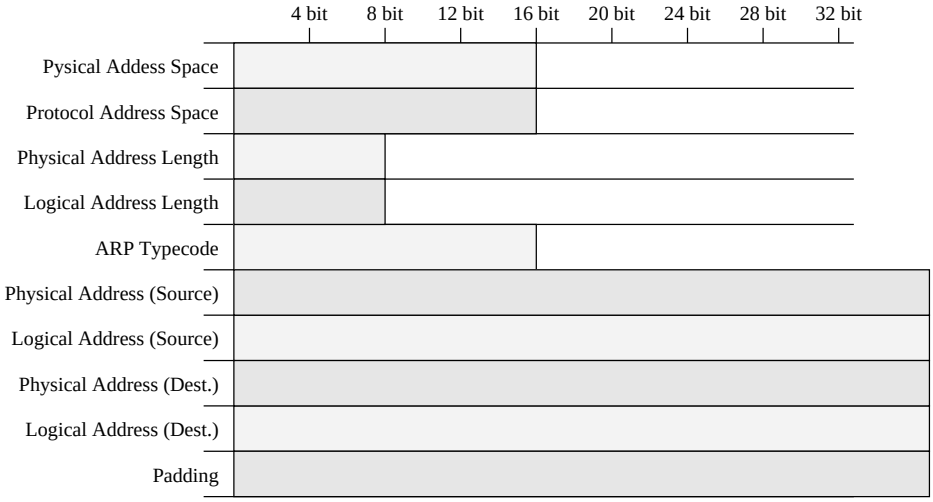

Abbildung 5.2: Aufbau des ARPv4 Headers

Tabelle 5.1: Beschreibung der Felder im ARPv4 Header

Feld	**Beschreibung**
Phy. Addr Space	Der Physical Address Space bestimmt die kleinste physikalische Addresse. Bei der Verwendung von MAC steht dort eine eins
Prot. Addr Space	Die Protocol Address Space gibt die kleinste Protokollnummer im Frame bekannt. Im Ethernet_II Frame ist es die `0x0800`
Phy. Length	Angabe der Länge einer Hardwareaddresse in Bytes. Bei MAC Addressen beinhaltet dieses Feld eine 6
Log. Length	Angabe der Länge einer logischen Addresse im Protokoll. Da IPv4 mit 32 bit Breiten Addressen arbeitet befindet sich bei der Verwendung von IPv4 in diesem Feld ein 4
Type	Dieses Feld entscheidet darüber ob es sich um ein ARP-Request oder um ein ARP-Reply handelt. Beim Request beinhaltet das Feld eine 1, beim Reply eine 2
Addresses	Diese beiden Felder sind in der Länge variabel. Die Länge wird in den beiden Length Felder entsprechend Vermerkt. Zu erst kommt die physikalische Addresse und danach die logische Addresse. Das ist sowohl beim Sender als auch beim Target so.

Die einzelnen Vorgänge des ARPv4's sind in den folgenden Abschnitten festgehalten. Sie beginnen mit dem ARP-Request bis hin zum ARP-Reply. Die folgenden Beispiele beziehen sich auf ein Ethernet.

5.2.1 ARP-Request

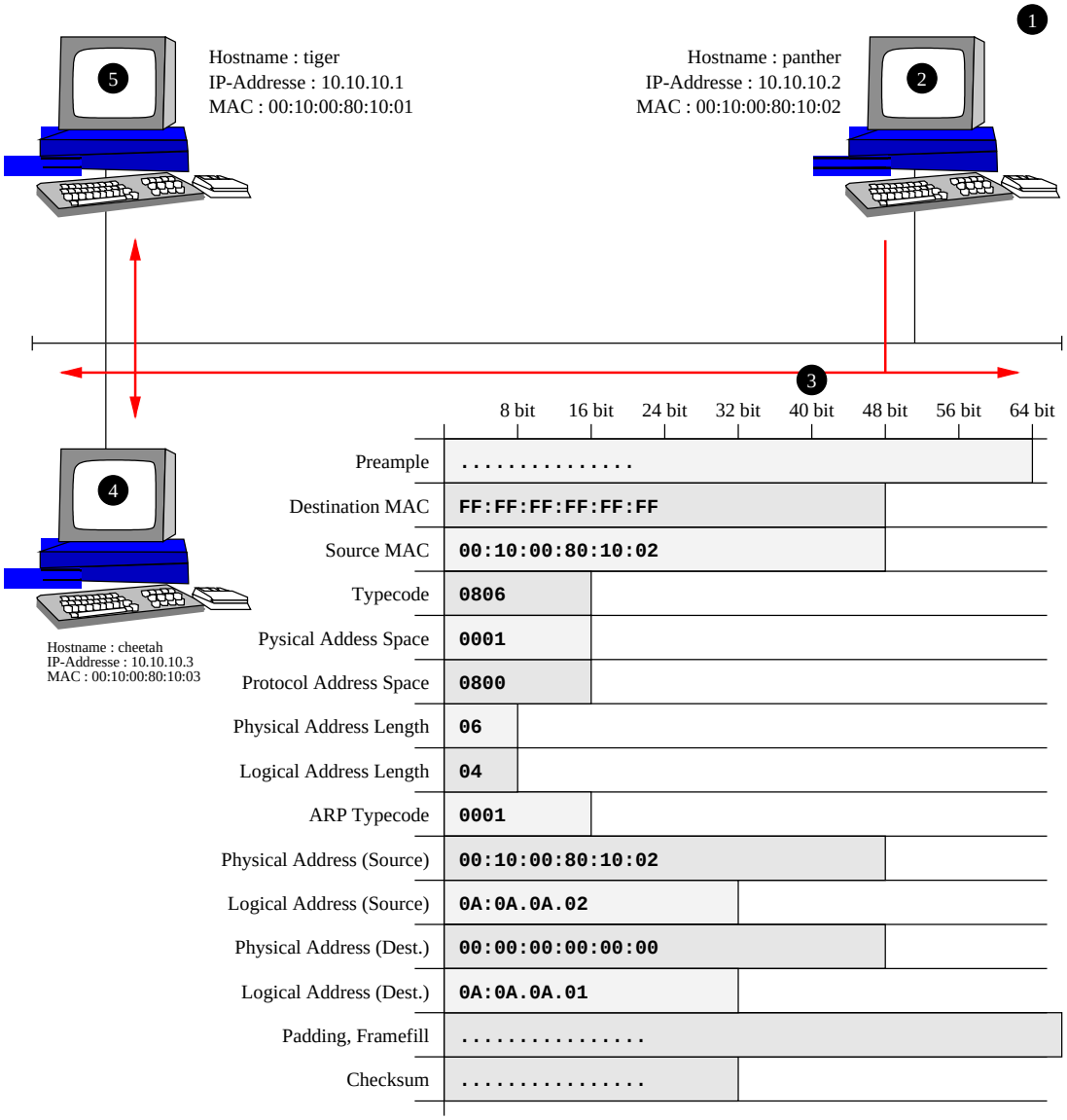

Abbildung 5.3: Ein ARPv4-Request

In der Grafik 5.3 haben die Nummern folgende Bedeutung :

1. Ein Benutzer will eine Verbindung vom Host PANTHER zum Host TIGER aufbauen. Welche Art von Verbindung soll erst einmal egal sein

2. Zur Komunikation benötigt PANTHER die MAC Addresse von TIGER. Er schaut in seinen ARP-Cache hinein sund sucht die MAC Addresse. Wir gehen davon aus, daß die Addresse nicht vorhanden ist. PANTHER muß also nach der MAC Addresse suchen

3. PANTHER bereitet nun ein ARP-Request vor und sendet diese als Broadcast auf das Netzwerk

4. Der Host CHEETAH bekommt nun diesen ARP-Request-Broadcast und analysiert diesen. Schnell merkt er, daß die *Logical Destination Address* nicht mit seinen IP-Addressen übereinstimmt. CHEETAH ignoriert den ARP-Request.

5. Der TIGER bekommt auch diesen ARP-Request-Broadcast. TIGER merkt das die *Logical Destination Address* mit einer seinen IP-Addressen überein stimmt. TIGER bereite im nächsten Abschnitt dann sein Reply vor.

5.2.2 ARP-Reply

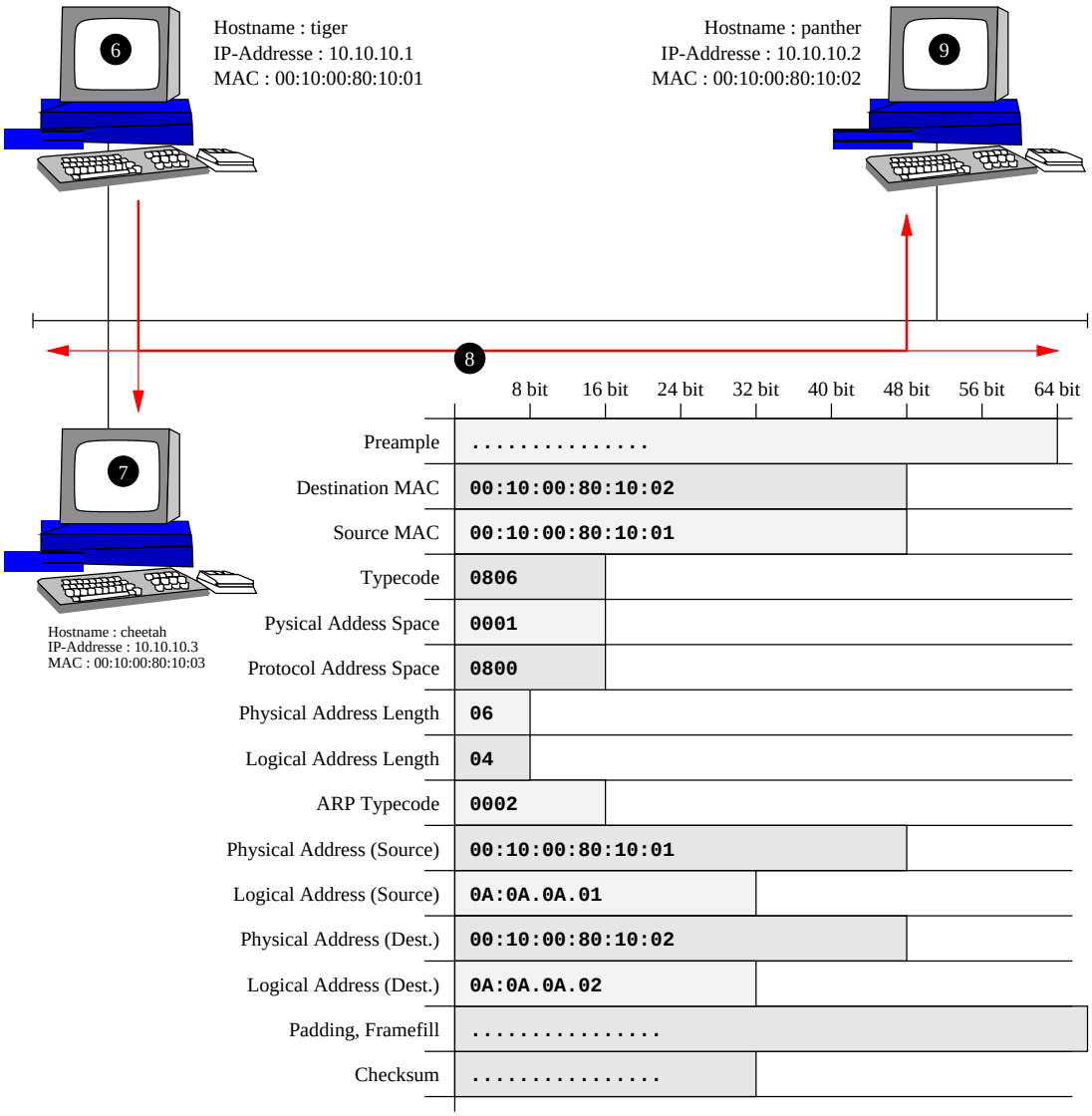

Abbildung 5.4: Ein ARPv4-Reply

In der Grafik 5.4 haben die Nummern folgende Bedeutung :

6. TIGER sendet nun den ARP-Request direkt an den PANTHER. Die MAC Addresse vom PANTHER

kennt er durch den APR-Request vom ihm.

7. CHEETAH bekommt auf einem BNC oder 10-Base-T die Antwort vom TIGER mit, da jedoch in der *Destination MAC* nicht seine MAC Addresse steht, ignoriert er den Frame
8. Der Frame wird gesendet
9. TIGER ist nun glücklich ! Er hat jetzt endlich die MAC Addresse vom PANTHER und kann mit der eigendlichen Komunikation anfangen

Kapitel 6

ANALYSE UND KONFIGURATIONSTOOLS

Die Analyse und die Konfiguration geht nur mit bereits Konfigurierte Hosts. Sorgen Sie dafür das alle Hosts im lokalem Netzwerk sich an pingen können. Ein pingen ist eine Art festzustellen ob ein Host korrekt konfiguriert wurde oder nicht. Zum pingen kann das Programm `ping` verwendet werden.

6.1 ping

Das Programm `ping` benutzt *ICMP* Nachrichten um die existenz eines Hostes in einem Netzwerk zu testen. Siehe hierzu das Kapitel ICMP im Internet Layer. Das Programm `ping` muß mit einer Zieladdresse oder einem Zielhostnamen, sofern verfügbar, aufgerufen werden. Jenach implementation gibt `ping` entweder `host is alive` oder ähnliche Meldungen zurück.

```
ping hostname oder ip
```

```
tiger: dozent $ ping falcon
PING falcon.hurst.pnet (10.65.13.1): 56 data bytes
64 bytes from 10.65.13.1: icmp_seq=0 ttl=255 time=0.283 ms
64 bytes from 10.65.13.1: icmp_seq=1 ttl=255 time=0.239 ms
64 bytes from 10.65.13.1: icmp_seq=2 ttl=255 time=0.218 ms
--- falcon.hurst.pnet ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 0.218/0.246/0.283 ms
tiger: dozent $
```

Bildschirmausschnitt 6.1.1: Beispiele von `ping` unter Linux

Hier wird auf dem Host TIGER ein `ping` zum FALCON ausgeführt. Danach wird er dreimal ge`ping`t und wird mittel CTRL-C abgebrochen.

Erscheint beim Pingversuch garnix mehr, dann ist die Zieladdresse falsch.

Escheint beim Pingen nur die erste Zeile : *PING .... (....): 56 data bytes* und sonst nix mehr, dann antwortet das Ziel nicht richtig.

Erscheint die Meldung *Network unreachable* fortwährend, dann ist die Route auf dem eigenen Rechner falsch. Siehe Routing.

6.2 arp

Das Programm `arp` dient zur Manipulation und zum Auslesen des kernelinternen ARP-Caches. Das Programm selbst ist auch in jeder TCP/IP Implementation vorhanden.

```
arp -a
arp -s hostname/-ip hardware_addr
arp -d hostname/-ip
arp -f file
```

Tabelle 6.1: Optionen zum Progamm `arp`

Option	Beschreibung
`-a`	Diese Option dient zur Anzeige des internen Kernelspeichers. Die Ausgabe des Programm unterscheidet sich von System zu System. Hierbei empfiehlt es sich die Hilfedatei sich anzulesen. Die man Page `arp` hilft bei den Abkürzungen sehr.
`-s`	ARP arbeitet dynamisch. Mit -s kann das verhalten geändert werden indem man eine Hardwareaddresse manuell einem Hostnamen oder einer IP zuordnet. TCP/IP startet dann keine ARP-Requests mehr und verwendet dann nur noch die selbst eingetragende Hardwareaddresse. Sinnvoll bei Hochsicherheitsserver.
`-d`	Mit -d kann ein statischer oder dynamischer Eintrag aus dem Cache manuell gelöscht werden
`-f`	Mit -f kann der Name einer ASCII Textdatei angegeben werden die IP Addressen und Hardwareaddressen auflistet. Diese Datei wird dann entsprechend eingelesen und die Einträge dort landen als statische Einträge im Kernelspeicher. Ebenfalls sehr sinnvoll bei Hochsicherheitsserver und Clients. Das Format dieser Datei ist recht simpel. Jede Zeile besteht aus zwei Spalten die jeweils durch ein Leerzeichen oder Tab getrennt sind. In der ersten Spalte steht der Hostname bzw. die Hostip und in der zweiten Spalte steht die Hardwareaddresse. Jedoch sollte man auch hier ein Blick in die man Page riskieren !
`-h`	Bei einigen Implmentationen existiert noch die Option -h für die Hilfe. Die Implementationen die mit -h nicht zurecht kommen geben die Hilfe preis, wenn man keine Optionen angibt.

Das folgende Szenario zeigt das Verhalten bei einer falschen Hardwareaddresse :

```
crow: dozent as root # arp -a
Device   IP Address               Mask      Flags   Phys Addr
------ -------------------- -------------- ----- ---------------
hme0   puma.hurst.pnet      255.255.255.255       00:00:b4:a6:00:9b
hme0   crow.hurst.pnet      255.255.255.255 SP    08:00:20:a6:a0:43
crow: dozent as root # ping shark.hurst.pnet
shark.hurst.pnet is alive
crow: dozent as root # arp -a
Device   IP Address               Mask      Flags   Phys Addr
------ -------------------- -------------- ----- ---------------
hme0   shark.hurst.pnet     255.255.255.255       00:00:b4:a6:7c:ee
hme0   crow.hurst.pnet      255.255.255.255 SP    08:00:20:a6:a0:43
crow: dozent as root # arp -s shark.hurst.pnet 00:10:20:30:40:50
crow: dozent as root # arp -a
Device   IP Address               Mask      Flags   Phys Addr
------ -------------------- -------------- ----- ---------------
hme0   crow.hurst.pnet      255.255.255.255 SP    08:00:20:a6:a0:43
hme0   shark.hurst.pnet     255.255.255.255 S     00:10:20:30:40:50
crow: dozent as root # ping shark.hurst.pnet
no answer from shark.hurst.pnet
crow: dozent as root # arp -d shark.hurst.pnet
shark.hurst.pnet (10.65.13.11) deleted
crow: dozent as root # arp -a
Device   IP Address               Mask      Flags   Phys Addr
------ -------------------- -------------- ----- ---------------
hme0   crow.hurst.pnet      255.255.255.255 SP    08:00:20:a6:a0:43
crow: dozent as root # ping shark.hurst.pnet
shark.hurst.pnet is alive
crow: dozent as root # arp -a
Device   IP Address               Mask      Flags   Phys Addr
------ -------------------- -------------- ----- ---------------
hme0   shark.hurst.pnet     255.255.255.255       00:00:b4:a6:7c:ee
hme0   crow.hurst.pnet      255.255.255.255 SP    08:00:20:a6:a0:43
crow: dozent as root #
```

Bildschirmausschnitt 6.2.1: Anwendung des `arp` Programms

Zuerst wird gezeigt das der Host SHARK sich nicht im ARP-Cache befindet. Er wird ange`ping`t, danach ist er im ARP-Cache. Nun wird für SHARK eine statische falsche Hardwareaddresse vergeben und nix geht mehr. Nach der Löschung der falschen Hardwareaddresse gehts wieder.

6.3 tcpdump

Das Programm `tcpdump` ist ein Netzwerkmonitor unter Linux.

17:56:36.156710 crow.hurst.pnet ¿ panther.hurst.pnet: icmp: echo request (DF) 17:56:36.156879 arp who-has crow.hurst.pnet tell panther.hurst.pnet 17:56:36.157118 arp reply crow.hurst.pnet is-at

8:0:20:a6:a0:43 17:56:36.157150 panther.hurst.pnet ¿ crow.hurst.pnet: icmp: echo reply

TEIL II

INTERNET LAYER (IP VERSION 4)

- Einführung
- Addressierung
  - Addressbereiche
  - Reservierte Addressen
- Das Internet Protocol
  - ToSlabel section:ipv4:ToS
  - Flagslabel section:ipv4:flags
  - Protocollabel section:ipv4:protocols
  - Optionslabel section:ipv4:options
- ICMP
  - Echo Request / Echo Reply
  - Destination Unreachable
  - Source Quench
  - Redirect
  - Time Exceeded
  - Parameter Problem
  - Timestamp / Timestamp Reply
  - Information Request / Information Reply
  - Domain Name Messages
- IGMP

Kapitel 7

EINFÜHRUNG

Wie schon erwähnt arbeitet der Internet Layer nicht mit MAC-Addressen sondern mit den IP-Addressen und es addressiert damit die Hosts. IP selbst ist ein sehr unzuverlässiges Protokoll, da es keinerlei Sicherungseinrichtungen besitzt. Der Sender kann nicht feststellen ob ein Datagram[1] angekommen ist oder nicht. Diese Funktionalität muß die nächst höhere Schicht mitbringen.

Im Internet Layer gibt es zwei Protokolle, zum einen das IPv4 und das IPv6, die wiederum einige kleinere Protokolle mitbringen. Wie widmen uns ersteinmal dem IPv4 mit seinen Unterprotokollen.

[1] `Datagram`-Daten die durch den Internet Layer laufen

Kapitel 8

ADDRESSIERUNG

Das Internet protocol verwendet zur Addressierung von Hosts und Netzwerken eine sogenannte IP-Addresse. Diese Addresse ist 32 bits Breit. Theoretisch können damit bis zu 4,3 Milliarden (4G) Addressen gebildet werden. Wie geschrieben ... theoretisch. Da sich ein Mensch die Addresse eines Hosts in der Form von $00100111001000011101010100010010_2$ ultra schlecht merken kann und auch die Form `2134839200` ist extrem ungeeignet. Deswegen werden die 32 bits in 4 mal 8 bits Pakete zusammen geschnuert. Jedes dieser kleinen Pakete nennt man Octet oder auch Quadrupel. Jede IP-Addresse besteht also aus 4 Octets. Jetzt errechnet man für jedes dieser Octets einen Dezimalwert. Man bekommt 4 Dezimalwerte im Bereich von 0 bis 255. Diese 4 Werte trennt man dann durch einen Punkt voneinander. Die Tabelle 8.1 soll dieses noch einmal verdeutlichen. Man muß folgendes beachten : **Die dezimale Schreibweise von IP-Addressen hat NICHTS aber auch absolut nix mit irgendwelchen anderen zusammenhängen zu tun.** Weder mit dem Namen eines Host nocht mit sonst irgendwelchen geschichten. **Die dezimale Schreibform ist NUR für uns dumme Menschen, um uns eine Addresse besser merken zu können !**

32 bit IP Addresse	00100111_2	00100001_2	11010101_2	00010010_2
4 mal 8 bit	1. Octet	2. Octet	3. Octet	4. Octet
Umwandlung nach Dezimal	39	33	214	18

Tabelle 8.1: Beispiel einer IP-Addressberechnung von Binäer ins Dezimale

8.1 Addressbereiche

Es gibt genormte Addressbereiche um die wir uns später noch genauer kümmern werden. Es sei ersteinmal nur soviel gesagt das die Entscheidung in welchen Bereich eine IP-Addresse fällt nur im 1. Octet liegt. Die möglichen Bereiche nennt man Klassen. In der Tabelle 8.2 sind die wichtigsten Klassen aufgelistet.

Klasse	1. Octet	Bereich von	Bereich bis	Beschreibung
A	$0xxxxxxx_2$	`0.0.0.0`	`127.255.255.255`	Normale Benutzung
B	$10xxxxxx_2$	`128.0.0.0`	`191.255.255.255`	dito.
C	$110xxxxx_2$	`192.0.0.0`	`223.255.255.255`	dito.
D	$1110xxxx_2$	`224.0.0.0`	`239.255.255.255`	Multicasting
E	$1111xxxx_2$	`240.0.0.0`	`255.255.255.255`	Experimentell

Tabelle 8.2: Kurze IP-Klassenliste

8.2 Reservierte Addressen

Die Klasse A Addresse `10.0.0.0` gibt es im Internet nicht. Wenn ein privates Netzwerk mit dem Internet komunizieren möchte sollte es die `10.0.0.0` Netzwerkaddresse tragen. Weil sonst gehts

nicht.

Kapitel 9

DAS INTERNET PROTOCOL

Das IP verwendet auch zur Kommunikation untereinander einen Header mit Informationen den IP vor jedes Datagram anfügt. Wenn die Daten vom NAL kommen weird der Header entsprechend ausgewertet und entfernt, die Nutzdaten werden dann nach oben gereicht. Wenn ein Datagram gesendet werden soll wird der Header vor das Datagram geschrieben und dann weiter zum NAL gereicht. Der eigendliche Inhalt der Nutzdaten ist für IP absolut uninteressant. Die Grafik 9.1 zeigt den Aufbau des Headers, die Beschreibungen der einzelnen Felder befindet sich in der Tabelle 9.1. Auch Beschrieben in der [RFC 791][1].

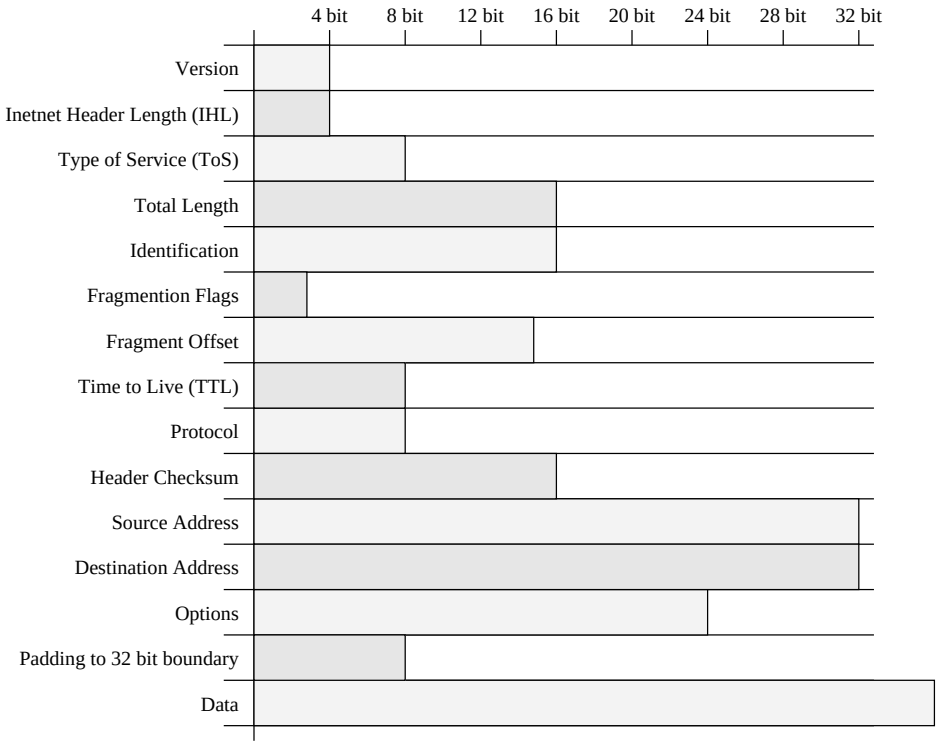

Abbildung 9.1: Der IPv4 Header

Tabelle 9.1: Inhaltsbeschreibung des IPv4 Headers

Feld	**Beschreibung**
Version	Gibt die Versionsnummer des IP Datagrams an. Zur Zeit gibt es nur entweder 4 oder 6 als gültige Werte

▼

[1] `RFC-791` Internet Protocol

▲

IHL	Das IHL[2] Feld beinhaltet die länge des IP Headers in 4 byte (32 bit) Schritten. Meist hat ein Header die länge von 24 Bytes, somit beinhaltet das IHL Feld meist eine 6 als Wert. Das Minimum ist 5
ToS	Das ToS[3] Feld setzt sich auf einzelnen Bits zusammen. Die Beschreibung des ToS Feldes befindet sich im Abschnitt 9.1 auf der nächsten Seite. Das ToS Feld regelt im groben die Behandlung des Datagrams in dessen Übertragungseigenschaften
Length	Das Length Feld gibt die Gesamtlänge des Datagrams in Bytes an. Der theoretische Maximalwert von 64kB wird jedoch von kaum einem Netzwerk unterstützt, wohingegen jede Implementation von IP einen mindestwert von 576 Bytes unterstützt. Weniger zu benutzen wäre etwas gefährlich, da Längen weniger 576 Bytes nicht vorgesehen wurden. Die Länge bezieht sich auf Nutzdaten und IP Header
Identification	Dient zur Identifikation des Datagrams, sofern es von IP fragmentiert wurde
Flags	Die Flags stehen im Zusammenhang mit der Fragmentierung von Datagramen. Das Feld besteht auch aus einzelnen bedeutsamen Bits. Eine Beschreibung befindet sich im Abschnitt 9.2 auf Seite 30
Fragment Offset	In diesem Feld wird die Position des Fragments im Datenstrom angegeben in 8 byte Schritten (64 bit). Sofern das Datagram fragmentiert werden musste
TTL	Das TTL[4] gibt die Lebensdauer eines Datagrams an. Das TTL Feld wird vom Sender auf einen bestimmten Wert gesetzt, meist 64 oder 128. Wenn das Datagram nun einen Router passiert dezimiert der Router dieses Feld um einen. Sofern ein Router ein Datagram mit einer TTL von NULL bekommt verwirft er dieses und sendet eine textslHost nichte erreichbar Meldung an den Sender
Protocol	In diesem Feld steht die Schlüßelnummer des Protokolls an den die Nutzdaten weitergereicht werden sollen. Die Beschreibung der Protokolle befindet sich im Abschnitt 9.3 auf Seite 30
Checksum	Prüfsumme nur des Headers. Vorsicht die muß jedesmal neuberechnet werden, weil ein Route das TTL Feld dezimiert
Source	Die 32 bit breite Quelladdresse des Ursprungssenders des Datagrams
Destination	Die 32 bit breite Zieladdresse des Empfaengers des Datagrams
Options	Eine ausführliche Beschreibung der Optionen siehe Abschnitt 9.4 auf Seite 31
Padding	Dieses Feld wird nur gebraucht damit der Header in ein 32 bit breites Raster passt. Es wird entsprechend der noch noetigen länge auf eine 32 bit Grenze mit irgendwelchen nutzlosen und vollkommen unsinnigen Werten gefüllt

[2] IHL-Internet Header Length
[3] ToS-Type of Service
[4] TTL-Time To Live

9.1 ToS

Das ToS ist für das Transferverhalten eines Datagrams verantwortlich. Es kann hier zum Beispiel gesteuert werden ob ein Datagram ohne Verzögerungen oder mit Vorrang behandelt werden soll. Die einzelnen Funktionalitäten sind in einzelne Bits aufgeteilt. Die Grafik 9.2 veranschaulicht diese Aufteilung. In der Tabelle 9.2 werden die einzelnen Bits erklärt.

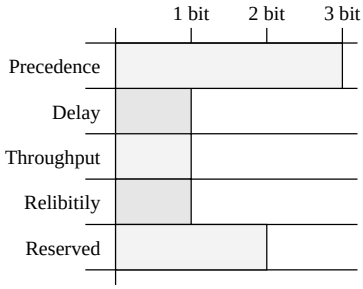

Abbildung 9.2: Type of Service Feld im IPv4

Tabelle 9.2: Beschreibung der Bits im Type of Service Feld vom IPv4

Feld	Beschreibung
Precedence	Das Precedence (Vorrang) Feld besteht aus 3 bits mit dem man 8 Zustaende definieren kann. Die 8 Zustaende werden auch voll ausgeschöpft. Die Tabelle 9.3 erläutert die einzelnen Zustände
Delay	Das Delay (Verzögerung) Feld bestimmt ob ein Datagram verzögert werden darf oder nicht. Der Normalfall sieht keine Verzögerung vor. Das Bit wird auf `1` gesetzt wenn das Datagram nicht verzögert werden darf. Das Datagram wird dann sofort weiter gesendet, sofern möglich. Ist meist bei zeitkritischen Anwendungen, wie RealVideo Anwendungen nötig
Throughput	Das Throughput ()
Relibility	Das Relibility (Zuverlässigkeit)
Reserved	Die Reserved (Reserviert) Bits sind für zukünftige Erweiterungen freigehalten worde, die jedoch in anbetracht von IPv6 eh nicht mehr genutzt werden

Tabelle 9.3: Precedence Bits im ToS des IPv4

Feld	Beschreibung
111_2	Local-Netzwerk Kontrolldaten im Datagram
110_2	Inter-Netzwerk Kontrolldaten im Datagram
101_2	Absolut höchst Kritisches Datagram
100_2	Höchst Kritisches Datagram
011_2	Kritisches Datagram
010_2	Noch mehr Vorrang

001_2	Vorrang gegenüber anderen Datagrammen
000_2	Kein Vorrang

9.2 Flags

Das Flagfeld ist nur 3 Bits breit. In den 3 Bits wird festgelegt ob ein Datagram Fragmentiert werden darf, oder nicht. Die Grafik 9.3 veranschaulischt das Feld und die Tabelle 9.4 beschreibt die einzelnen Inhalte.

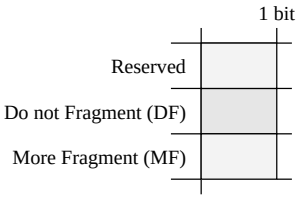

Abbildung 9.3: Flags Feld im IPv4

Tabelle 9.4: Beschreibung der Bits im Flags Feld vom IPv4

Feld	Beschreibung
Reserved	Reserviertes Bit. Muß NULL sein
DF	Das DF (Don't Fragment) Bit bestimmt ob ein Datagram fragmentiert werden darf. Ist das Bit gesetzt darf nicht fragmentiert werden
MF	Das MF (More Fragment) ist gesetzt wenn es sich um ein fragmentiertes Datagram handelt. Ist das Bit nicht gesetzt wird damit angezeit das dieses Datagram das letzte Datagram des Fragments ist

9.3 Protocol

Das Protocol Feld bestimmt an welches Protokoll letztendlich die Nutzdaten weitergereicht werden müssen. Jedes Betriebssystem das TCP/IP unterstützt beherbergt eine Datei mit dem Namen protocols in der die Protokolle aufgelistet werden. Fehlt in dieser Datei ein Eintrag kann auch das Protokoll nicht verwendet werden. Die Tabelle 9.5 zeigt eine Liste der Standariesierten Protokoll Nummern.

Tabelle 9.5: Protokollnummern im Feld Protocol vom IPv4

Feld	Beschreibung
0	IP
1	ICMP
2	IGMP
3	GGP[5]

[5] GGP-Gateway-Gateway Protocol

▲

6	TCP
12	PUP[6]
17	UDP
22	IDP - Keiner weiss was genaues ...
255	RAW - Roh Daten

9.4 Options

Das Optionsfeld ist variabel. Das soll heissen das es je nach Option eine unterschiedliche Länge sowie unterschiedlichen Inhalt besitzt. Die einzelnen Aufschluesselungen werden noch beschrieben ! Jedoch aus Zeit gründen ersteinmal unbeachtet.

[6] PUP-PARC universal packet protocol

Kapitel 10

ICMP

Das ICMP[1] ist ein Protokoll innerhalb des Internet Layers und gehoert mit zu TCP/IP. Das Protokoll sowie die Funktionalitaeten werden automatisch aktiviert und bereitgestellt, eine Deaktivierung ist nicht möglich.

Das ICMP ist im IPv4 sowie auch im IPv6 vorhanden. Im IPv6 wird es ICMPv6 genannt da es sich von ICMPv4 unterscheidet. Das ICMP benoetigt das IP um Informationen zwischen Hosts auszutauschen. Zum Beispiel wenn ein Datagram sein Ziel nicht erreichen kann oder wenn z.B. das TTL abgelaufen ist oder zum Austausch von Zeiten im Netzwerk, etc. Bis auf eine Außnahme verwendet eigendlich keine Anwendung ICMP direkt, bis auf `ping`. Nachzulesen in der [RFC 792][2].

Der ICMP Header hat annaehernt immer den gleichen Aufbau, je nach Typ der Nachrich veraendern sich die nachfolgenden Feldinhalte. Die Grafik 10.1 zeigt den unveraenderlichen Header. Die Tabelle 10.1 schluesselt die einzelnen Positionen auf.

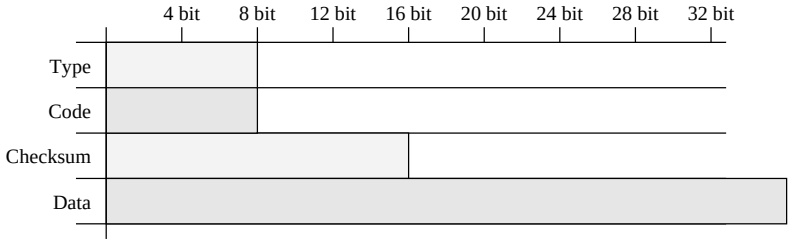

Abbildung 10.1: Grundheader des ICMPv4

Tabelle 10.1: Inhaltsbeschreibung des Grundheaders von ICMPv4

Feld	**Beschreibung**
Type	Gibt den Typ der Nachricht an. Siehe Tabelle 10.2
Code	Inhalt ist abhaengig vom Typ der Nachicht
Checksum	Pruefsumme der ICMP Nachricht. Gebildet aus dem Einer-Komplement der Summe der Einer-Komplemente
Variabel	Inhalt und länge ist abhaengig vom Typ der Nachicht

Tabelle 10.2: Typen von ICMPv4

Feld	**Beschreibung**
0	Echo Reply
3	Destination Unreachable

▼

[1] `ICMP`-Internet Control Message Protocol

[2] `RFC-792` Internet Control Message Protocol

▲

4	Source Quench
5	Redirect
8	Echo Request
11	Time Exceeded
12	Parameter Problem
13	Timestamp
14	Timestamp Reply
15	Information Request
16	Information Reply

10.1 Echo Request / Echo Reply

Ein Echo Reply wird versendet, wenn der TCP/IP Stack einen Echo Request (8) empfängt. Das Programm `ping` verwendet Echo Request Nachrichten um die Verfügbarkeit eines Hosts zu ermitteln. Dabei zählt `ping` für jedes Paket die Sequenznummer hoch, die beim Echo Reply wieder zurückgegeben wird. `ping` muß auf jedem TCP/IP fäehigen Betriebssystem verfügbar sein, weil es bestandteil der RFC ist. Das Codefeld ist 0.

Die Grafik 10.2 zeigt den Aufbau der Headers fuer den Echo Reply sowie fuer den Echo Request Typ der ICMP Nachricht. In der Tabelle 10.3 sind die Felder für den Echo Reply entsprechend aufgeschlüsselt.

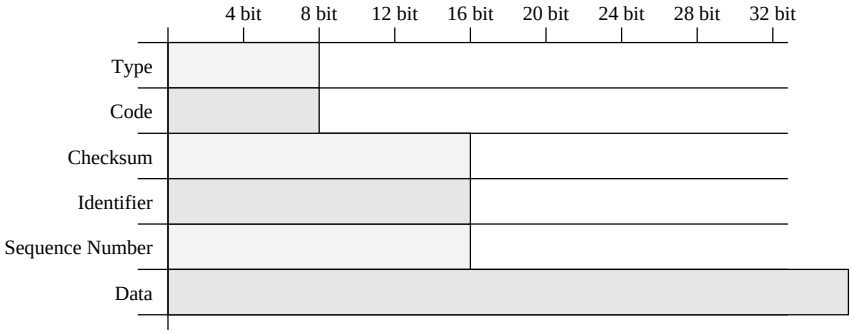

Abbildung 10.2: Echo Reply / Echo Request Header des ICMPv4

Tabelle 10.3: Inhalte des Echo Reply / Request Headers von ICMPv4

Feld	Beschreibung
Type	Steht Fest auf `0` bzw. `8` beim Request
Code	Ist immer 0
Checksum	Pruefsumme der ICMP Nachricht
Identifier	Identifikationsnummer dieser `ping` Session
Sequence	Sequenznummer dieser Nachricht
Data	Irgendwelche Daten, bzw. kopie der Daten beim Echo Request

10.2 Destination Unreachable

Dieser Nachrichtentyp wird verwendet wenn ein Datagram nicht ausgeliefert werden konnte, weil der Host bzw. ein Netzwerk nicht erreichbar ist (siehe Routing). Aber auch Hosts selbst können sich dieses Typs bedienen indem sie dem Sender berichten das ein Protokoll bzw. ein Port nicht erreichbar bzw. nicht installiert ist. Intern verwendet TCP/IP auch noch die Nachricht um Fehler bei der Fragmentierung zu melden.

Die Grafik 10.3 zeigt den Aufbau des Headers fuer diesen Typ der ICMP Nachricht. In der Tabelle 10.4 sind die Felder entsprechend ausgeschlüsselt.

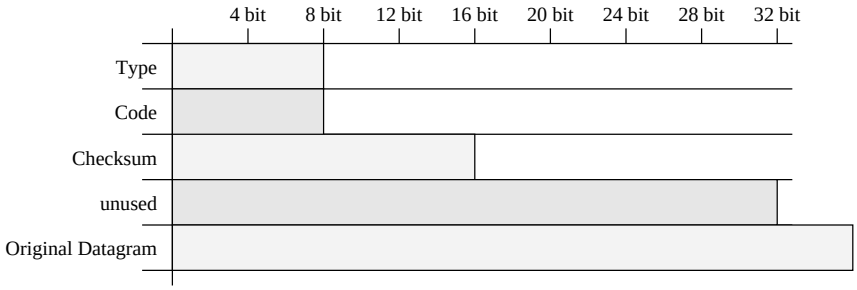

Abbildung 10.3: Destination Unreachable Header des ICMPv4

Tabelle 10.4: Inhalte des Destination Unrechable Headers von ICMPv4

Feld	Beschreibung
Type	Steht Fest auf 3
Code	Inhalt ist in Tabelle 10.5 aufgeschlüsselt
Checksum	Pruefsumme der ICMP Nachricht
unused	Unbenutzt
IH + Datagram	Internet Header inklusive 64 bits des orginalem Datagram

Tabelle 10.5: Codetypen des Destination Unreachable von ICMPv4

Feld	Beschreibung
0	Net Unreachable. Ziel Netzwerk ist nicht erreichbar
1	Host Unreachable. Ziel Host nicht erreichbar
2	Protocol Unreachable. Protokoll wird nicht unterstützt
3	Port Unreachable. Portnummer (siehe TCP/UDP) ungültig
4	Fragmentation benötigt aber kein DF Flag in IPv4→Flags gesetzt
5	Source route failed. Quelle nicht erreichbar

10.3 Source Quench

Eine Source Quench Nachricht wird vom Zielhost bzw. von einem Gateway aus an den Absender der Daten gesandt wenn auf dem Zielhost bzw. dem Gateway so viel Verkehr ist das dieser das Datagram nicht bearbeiten konnte. Der sendene Hosts muß das Datagram noch mal senden und er sollte die Datentransfergeschwindigkeit verringern.

Die Grafik 10.4 zeigt den Aufbau des Headers fuer diesen Typ der ICMP Nachricht. In der Tabelle 10.6 sind die Felder entsprechend ausgeschlüsselt.

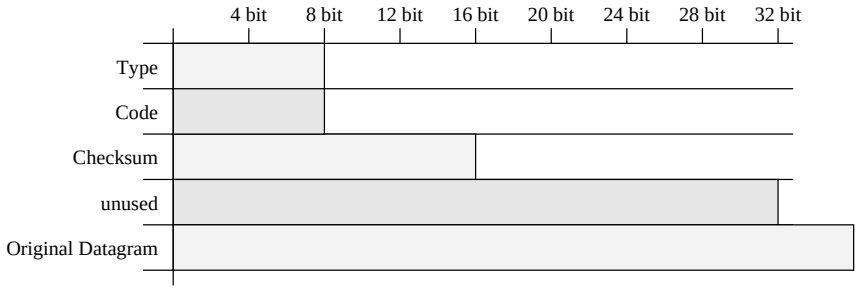

Abbildung 10.4: Source Quench Header des ICMPv4

Tabelle 10.6: Inhalte des Source Quench Headers von ICMPv4

Feld	**Beschreibung**
Type	Steht Fest auf 4
Code	Ist immer 0
Checksum	Pruefsumme der ICMP Nachricht
unused	Unbenutzt
IH + Datagram	Internet Header inklusive 64 bits des orginalem Datagram

10.4 Redirect

Eine Redirection (Umleitung) wird im folgenden Fall versandt: Ein Gateway (G1) bekommt ein Datagram fuer einen Host in Netzwerk X. Wenn das Gateway G1 feststellt das dieses Datagram an ein weiteres Gateway G2 versandt werden soll und dieses Gateway G2 befindet sich im gleichen Netzwerk wie auch der Absender, dann wird dem Sender eine Redirection Nachricht gesendet. Somit kann der Host fuer das Netzwerk X das andere Gateway G2 benutzen. Das verringert den Verkehr im Netzwerk. Das Datagram läuft ja sonst zweimal über die gleiche Leitung.

Die Grafik 10.5 zeigt den Aufbau des Headers fuer diesen Typ der ICMP Nachricht. In der Tabelle 10.7 sind die Felder entsprechend ausgeschlüsselt.

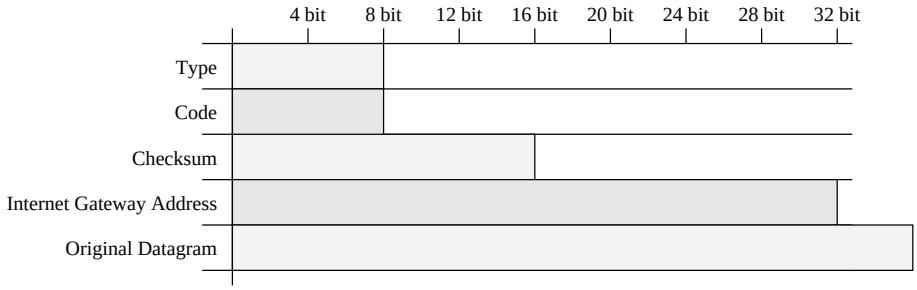

Abbildung 10.5: Redirect Header des ICMPv4

Tabelle 10.7: Inhalte des Redirect Headers von ICMPv4

Feld	Beschreibung
Type	Steht Fest auf 5
Code	Inhalt ist in Tabelle 10.8 aufgeschlüsselt
Checksum	Pruefsumme der ICMP Nachricht
Gateway	Das eigendliche Gateway das der Host benutzen sollte

Tabelle 10.8: Codetypen des Redirect von ICMPv4

Feld	Beschreibung
0	Umleitung der Datagrame für ein Netzwerk
1	Umleitung der Datagrame für einen Host
2	Umleitung der Datagrame für ein Netzwerk und mit speziellen ToS
3	Umleitung der Datagrame für einen Host und mit speziellen ToS

10.5 Time Exceeded

Eine Time Exceeded Nachricht wird versendet wenn das ankommende Paket eine TTL von NULL hat. Oder wenn ein Datagram Fragment nicht richtig zusammen gesetzt werden kann. Das betroffende Datagram wird dabei vernichtet.

Die Grafik 10.6 zeigt den Aufbau des Headers fuer diesen Typ der ICMP Nachricht. In der Tabelle 10.9 sind die Felder entsprechend ausgeschlüsselt.

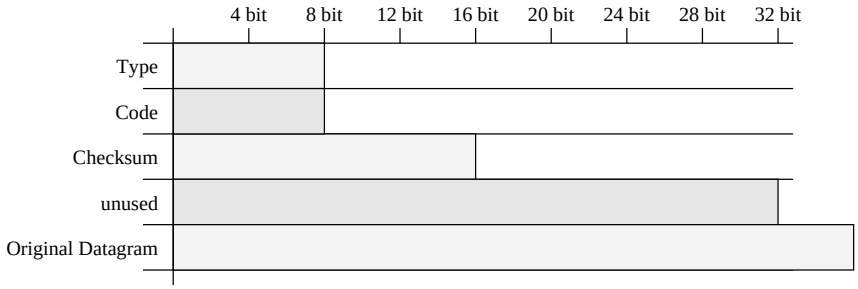

Abbildung 10.6: Time Exceeded Header des ICMPv4

Tabelle 10.9: Inhalte des Time Exceeded Headers von ICMPv4

Feld	Beschreibung
Type	Steht Fest auf `11`
Code	Inhalt ist in Tabelle 10.10 aufgeschlüsselt
Checksum	Pruefsumme der ICMP Nachricht
unused	Unbenutzt
IH + Datagram	Internet Header inklusive 64 bits des orginalem Datagram

Tabelle 10.10: Codetypen des Time Exceeded von ICMPv4

Feld	Beschreibung
0	TTL abgelaufen
1	Fragmentierung unvollständig

10.6 Parameter Problem

Wenn ein Host oder Gateway ein Datagram bekommt versendet er eine Parameter Problem Nachricht wenn er im IP Header einen Fehler oder unklarheiten findet. Das Feld Pointer beschreibt dann die Feldnummer wo das Problem auftrat. Das betroffende Datagram wird vernichtet.

Die Grafik 10.7 zeigt den Aufbau des Headers fuer diesen Typ der ICMP Nachricht. In der Tabelle 10.11 sind die Felder entsprechend ausgeschlüsselt.

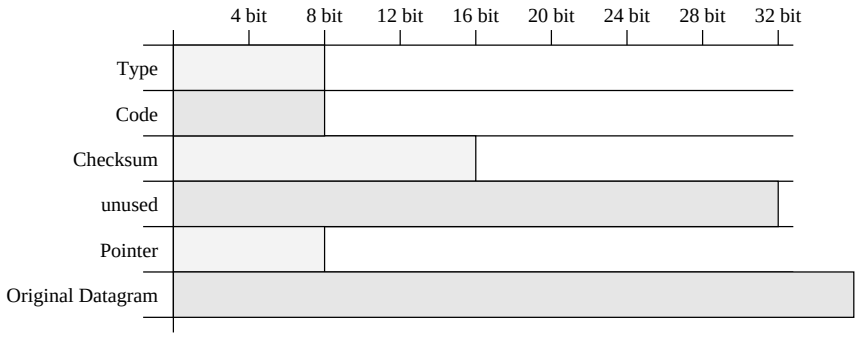

Abbildung 10.7: Parameter Problem Header des ICMPv4

Tabelle 10.11: Inhalte des Parameter Problem Headers von ICMPv4

Feld	Beschreibung
Type	Steht Fest auf `12`
Code	Inhalt ist in Tabelle 10.12 aufgeschlüsselt
Checksum	Pruefsumme der ICMP Nachricht
Pointer	Beinhaltet die Feld Position des Fehlers (Feld 1 == ToS)
unused	Unbenutzt
IH + Datagram	Internet Header inklusive 64 bits des orginalem Datagram

Tabelle 10.12: Codetypen des Parameter Problem von ICMPv4

Feld	Beschreibung
0	Pointer zeigt auf das Problem

10.7 Timestamp / Timestamp Reply

Mit diesem Nachrichen Typ kann man die Geschwindigkeit des TCP/IP Stacks und die Leitungsgeschwindigkeit überprüfen. Aber auch Zeitsynconisationen sind hier möglich, jedoch kommen für diesen Zweck andere Programme zum Einsatz. Gemessen wird hier in Millisekunden.

Die Grafik 10.8 zeigt den Aufbau des Headers fuer diesen Typ der ICMP Nachricht. In der Tabelle 10.13 sind die Felder entsprechend ausgeschlüsselt.

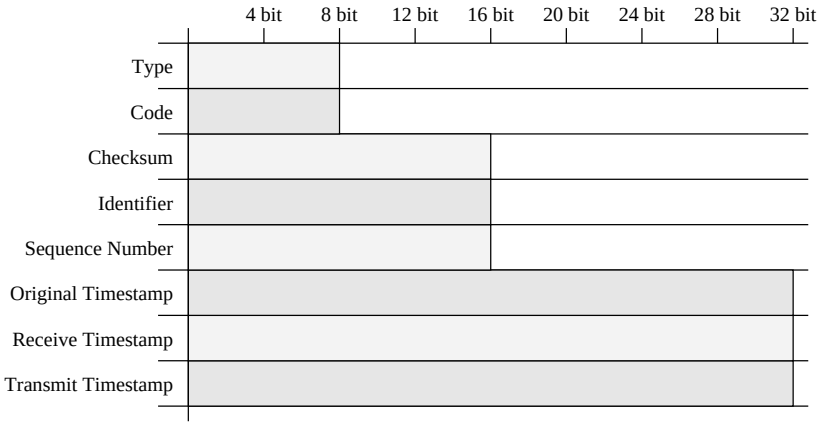

Abbildung 10.8: Timestamp Header des ICMPv4

Tabelle 10.13: Inhalte des Timestamp Headers von ICMPv4

Feld	Beschreibung
Type	Steht Fest auf `13` bzw `14` beim Reply
Code	Ist immer 0
Checksum	Pruefsumme der ICMP Nachricht
Orginate	Zeitstempel vom Sender beim Versandt
Receive	Zeitstempel vom Empfänger beim Empfang
Transmit	Dauer der Übertragung

10.8 Information Request / Information Reply

Mit diesem Typ kann festgestellt werden welchem Netzwerk der Host angehört. Im ernstfall wird es jedoch kaum verwendet.

Die Grafik 10.9 zeigt den Aufbau des Headers fuer diesen Typ der ICMP Nachricht. In der Tabelle 10.14 sind die Felder entsprechend ausgeschlüsselt.

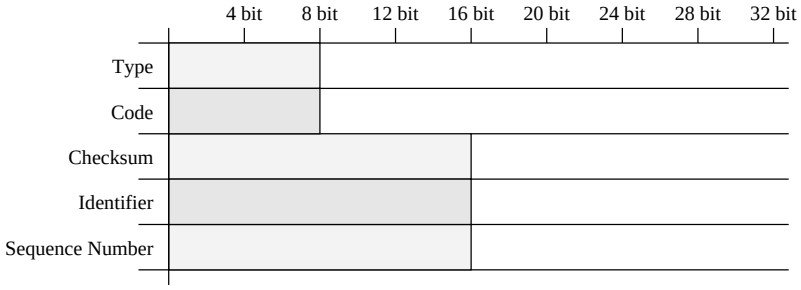

Abbildung 10.9: Information Request/Reply Header des ICMPv4

Tabelle 10.14: Inhalte des Information Request Headers von ICMPv4

Feld	Beschreibung
Type	Steht Fest auf `15` bzw. `16` für Reply
Code	Ist immer NULL
Checksum	Pruefsumme der ICMP Nachricht
Identifier	Entspricht wie Echo
Sequence	Entspricht wie Echo

10.9 Domain Name Messages

Diese ICMP Nachricht gehört zu den neueren ICMP Nachrichten. Die [RFC 1788][3] wurde im April 1995 freigegeben. Die Aufgabe der ICMP Nachricht besteht darin die DNS Domainnamen auf Anfrage bekannt zu geben.

Es gibt zum einen den Request und den Reply. Beine Nachrichten unterscheiden sich leicht von Ihrem Aufbau her. Daher gibt es nur eine Zeichnung. Die letzten zwei Felder jedoch existieren nur im Reply.

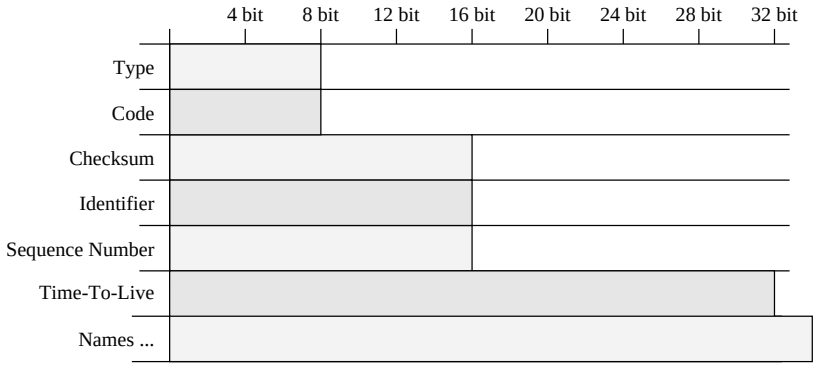

Abbildung 10.10: Domain Name Messages Request/Reply Header des ICMPv4

Tabelle 10.15: Inhalte des Domain Name Message Request/Reply Headers von ICMPv4

Feld	Beschreibung
Type	Steht fest auf 37 wenn es ein Request ist, bzw. auf 38 für den Reply
Code	Ist immer NULL
Checksum	Prüfsumme der ICMP Nachricht
Identifier	In diesem Feld wird eine Zufallsnummer auf dem zusendenen Host benutzt. Da mehrere Prozesse zur gleichen Zeit ein ICMP DNM Request senden könnten, muß der TCP/IP Stack des Systems die ankommenden Antworten richtig zuweisen können. Dazu dient dieses Feld
Sequence	Jede Anfrage eines Prozesses bekommt eine Sequenznummer die hochgezählt wird. Somit weiß nun auch der Prozeß selbst um welche Antwort es sich handelt
Time-To-Live	Dieses Feld ist nur im Reply Header zu finden. Es gibt die Zeit in Sekunden an, die dieser Eintrag maximal im Cache des Anfragenden verleiben darf. Danach sollte der anfragende Host noch einmal Fragen
Names ...	Hier sind alle Domainnamen aufgelistet. Die MTU entscheidet über die Anzahl der Domainnamen. Ist die MTU zu klein werden nicht alle gesendet

Für diese funktionalität muß auf den entsprechenden Betriebssystem ein ICMP Domainname Server

[3] RFC-1788 ICMP Domain Name Messages

aufgesetzt werden. Diese Funktion wird nicht vom TCP/IP Stack übernommen.

10.9.1 Funktionsweise und Gebrauch

Die ICMP Domain Name Message hat nur eine bestimmte Aufgabe, nämlich einem anfragendem Host seine Domains zu senden. Dabei sendet der Client ein ICMP Domain Name Message Request an den ICMP DNS Server bzw. er sendet die Nachricht als Broadcast in das Netzwerk. Wird die Nachricht via Broadcast gesendet, dann muß sich der ICMP DNS Server im gleichen physikalischen und logischen Netzwerk befinden. Der ICMP DNS Server empfängt die Nachricht und ermittelt nun die Quell IP Adresse aus dem IP Header. Mit dieser IP Adresse fragt der ICMP DNS Server seine Revers-Lookup-Zone und sendet dem Client die Domainnamen wieder zurück die er aus der Zone filtern konnte.

Und das Tolle ? Nun man kann sich die Konfiguration des Domainnamens auf dem Client komplett sparen, wenn man beim Hochfahren ein Request startet. Weiterhin ist diese Methode wesendlich schneller als alles andere was es noch so gibt.

10.9.2 C Struktur

Als Basis kann die ICMP Struktur `icmp` aus dem Header `netinet/ip_icmp.h` genommen werden. Da der Typ erst später kam muß man wohl oder übel die Struktur selbst schreiben :

Quelltext 10.9.1 C Struktur der ICMPv4 Domain Name Messages

```
struct icmp_dnm {
        uchar_t type;     /* type of message, 37 or 38 */
        uchar_t code;     /* type sub code is ever 0 */
        ushort_t cksum;   /* ones complement cksum of struct */
        ushort_t id;      /* process identifierer */
        ushort_t seq;     /* sequence */
        uint_t ttl;       /* time-to-live in cache */
        char names[1];    /* begin of names */
};
```

Kapitel 11

IGMP

Das IGMP[1] wird eingesetzt um mehrere Hosts einer virtuellen Addresse zuzuweisen. Es besteht die Möglichkeit über die D Classen Addressen Gruppen zudefinieren. Mit dieser Technik wird ein Datagram nur einmal ausgesendet und automatisch an mehrere Hosts verteilt. Zur Zeit werden einige News-Server auf Groups umgestellt. Das Protokoll jedoch erfuhr in den letzten Jahren einige Renovierungen. [RFC 988][2] seit July 1986 wurde von [RFC 1112][3] im August 1989 abgeloest bis es schliesslich von [RFC 2236][4] im November 1997 vollständig abgelöst wurde. Wir werden später nocheinmal auf Multicasting eingehen.

[1] `IGMP`-Internet Group Management Protocol
[2] `RFC-988` Host Extension for IP Multicasting
[3] `RFC-1112` Host Extension for IP Multicasting
[4] `RFC-2236` Internet Group Management Protocol, Version 2

TEIL III

TRANSPORT LAYER

- Einführung
- Portaddressen
- UDP
- TCP
  - TCP Optionslabel section:tcp-options
- Funktionsweise von TCP
  - Verbindungsaufbau
  - Erfolglose Verbindungsaufbaulabel sec:conref
  - Einfache Datenaustausch
  - Komplexer Datenaustausch
  - Automatische Fehlerbehebung
  - Beenden einer Verbindung
  - Ein kompletter Datenaustausch

Kapitel 12

EINFÜHRUNG

Wie nun die Daten an die Hosts weiter verteilt werden wissen wir jetzt, dank des Internet Layers. Wie gelangen jedoch die Daten an die richtige Anwendung ? Auf einem Client können ja mehrere Programme gleichzeitig laufen. Denken Sie nur an ein gestarteten Webbrowser (Netscape) und vielleicht ein parallellaufender FTP Download. Wie bekommt jetzt nun der Browser seine Daten und wie gelangen die Downloaddaten auch an den FTP Client ? Diese Frage muss wohl auch den beschäftigt haben der TCP/IP entworfen hat. Er kam auch auf eine Lösung. Die Lösung des Problems lautet : Portaddressen.

Kapitel 13

PORTADDRESSEN

Ein Port ist eine Nummer. Die Nummer kann im Bereich von 0 bis 65535 liegen. Eine Portnummer ist demzufolge also 16 bits Breit. Die ersten 1024 Portaddressen sind standarisiert, und sollten auf keinen Fall geändert werden. Welche Portnummern der aktuelle Rechner unterstützt ist in der /etc/services festgehalten.

Mal ein kleiner Beispiel: Auf Host A wird ein Web-Server (Apache) installiert. Der Apache öffnet einen Datenkanal auf Port 80, und horcht solange am Port bis was kommt. Auf dem Host B wird ein Webbrowser (Netscape) gestartet. Man gibt nun die Addresse des Hosts A in die Addresszeile ein. Der Netscape versucht nun Verbindung zum Host A, wo der Apache läuft, kontakt aufzunehmen. Der Netscape nimmt den Port 80, weil dieser Standard ist für HTTP, und weil Sie keinen anderen gewählt haben. Am Host A kommt jetzt nun die Anforderung für Port 80 an, da sich noch der Apache an diesem Port befindet, bekommt er die Nachricht zugestellt. Jetzt kann er entsprechend antworten. Stellt sich die Frage wohin der Apache Antworten soll ? Der Apache kann nicht auch auf Port 80 seine Antwort senden, da es ja sein kann das auf dem Host B auch ein Server läuft (Multitasking). Die Antwort ist einfach: Bevor der Netscape seine Anforderung raussendet vergibt der TCP/IP Stack eine zur Zeit noch freie Portnummer. Diese Portnummer steht weder in der /etc/services noch wird diese gerade benutz. Diese willkürlich vergebene Portnummer wird nun der Anforderung einfach mitgesendet. Jetzt weiss der Apache an welchen Port er seine Antwort zu senden hat. Somit können mehrere Netscapes am gleichen Host surfen aber an verschiedenen Stellen, und alle Browsers bekommen auch die Daten die sie benötigen. Wenn der Browser mit seiner Anfrage fertig ist schließt er den Kanal, somit wird auch der willkürlich vergebene Port wieder freigegeben. Aber wie gehabt, die Grafik 13.1 auf der nächsten Seite sagt mehr als 2^{10} Worte ...

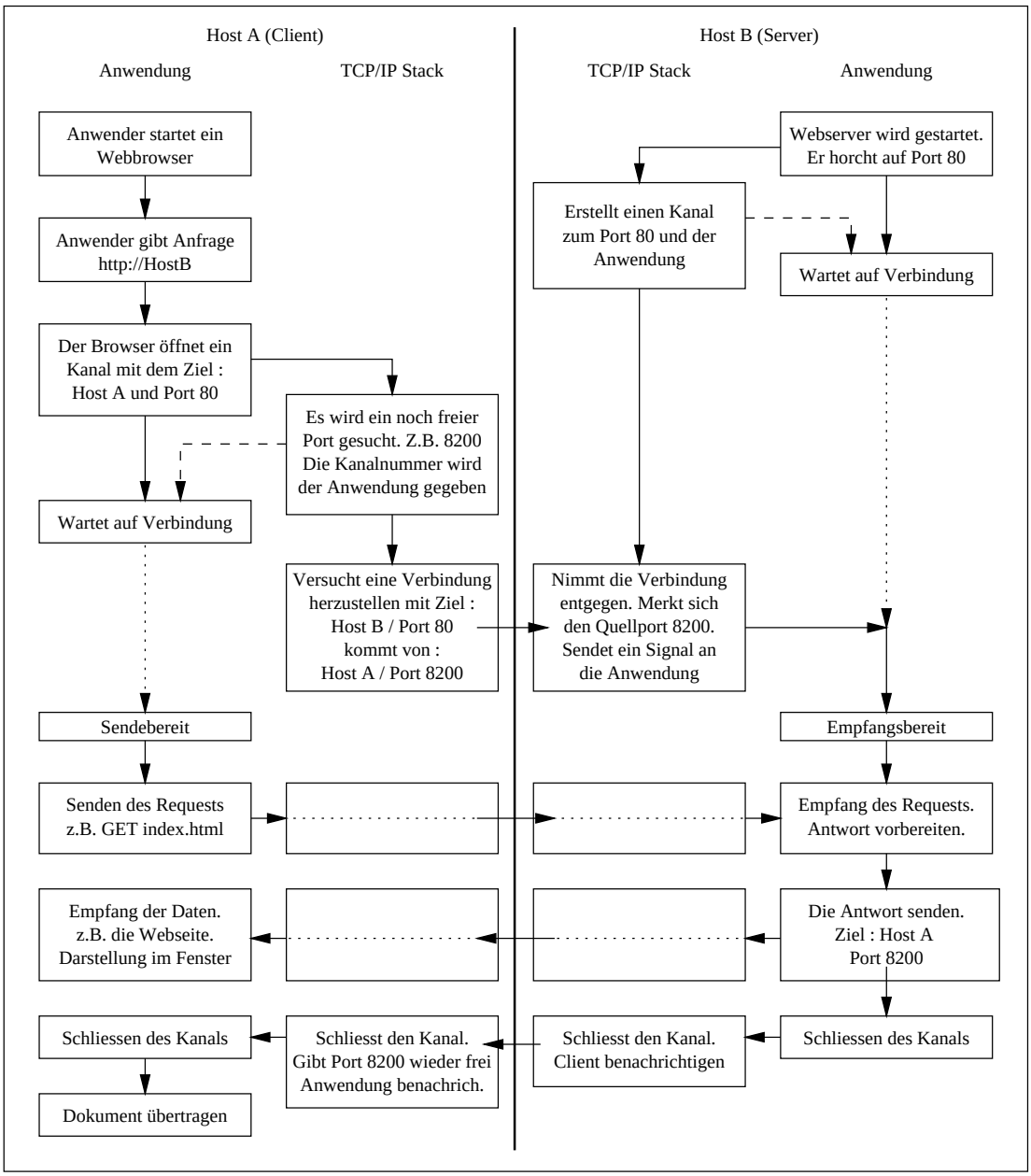

Abbildung 13.1: Ein Beispiel zur Portvergabe

Portaddressen existieren sowohl im UDP als auch im TCP, jedoch haben die Portaddressen miteinander nichts zu tun. Das soll bedeuten das es unter UDP 65536 Ports, und unter TCP auch 65536 Ports gibt. Insgesamt gibt es also 121172 mögliche Ports. Das hört sich viel an, ist jedoch nicht so viel wie wir noch sehen werden.

Kapitel 14

UDP

Das UDP[1] ist das kleinste unter den Transportprotokollen. Es besitzt keine Sicherung der Übertragung. Die Anwendung muß sich selbst um die Sicherung kümmern. UDP ist jedoch durch die extreme Einfachheit sehr schnell. Mit UDP sind auch Broadcasts möglich. Die Daten die durch UDP versendet werden nennt man Pakete, wobei es auf der Anwendungsschicht dann Nachrichten sind. UDP ist in [RFC 768][2] definiert. Die Grafik 14.1 und die Tabelle 14.1 zeigen den simplen Aufbau von UDP.

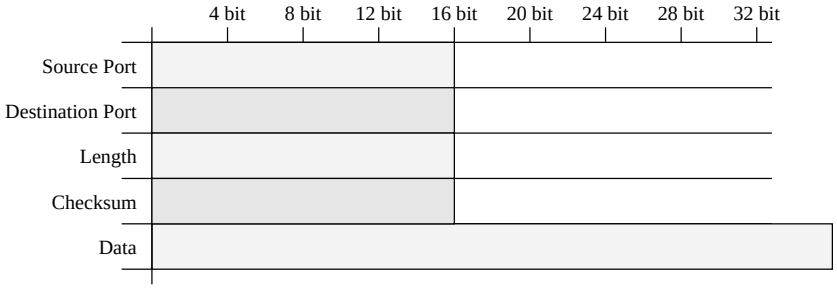

Abbildung 14.1: UDP Header

Tabelle 14.1: Beschreibung des UDP Headers

Feld	**Beschreibung**
Source Port	Nummer des Quellports des Senders der Nachricht
Destination Port	Nummer des Zielports des Empfängers
Length	Länge des Pakets inklusive des Headers in Bytes
Checksum	Prüfsumme der Nachricht

[1] UDP-User Datagram Protocol

[2] RFC-768 User Datagram Protocol

Kapitel 15

TCP

Das TCP[1] ist das aufwendigste Protokoll. Für die Anwendung jedoch das einfachste. Die Daten die durch TCP laufen nennt man **Segmente**. Das TCP bietet unabhängig der Anwendung die folgenden Leistungen :

Einfachste Programmierung von Seiten der Anwendung durch streams Das TCP bedient sich der einfachsten Elemente, wie lesen, schreiben, oeffnen, und schliessen der Verbindung. Wobei das lesen und schreiben genauso funktioniert als wenn man aus einer Datei lesen bzw. in ein Datei schreiben würde. Zwar heissen die Funktionen der Programmiersprache anderst aber das Prinzip ist das gleiche

Sichere Übertragungen und automatische Fehlerbehebungen Das TCP erkennt automatisch einen eventuellen Verbindungsabbruch und versendet seine zusendenen Daten neu. Der Empfänger erkennt eventuell doppelt versandte Segmente und verwirft die bereits erhaltenen. Bevor jedoch irgendwelche Daten ausgetauscht werden fragt der kontaktsuchende Host erst einmal nach ob der gegenüber überhaupt da ist

Automatische Flußkontrolle Das TCP regelt automatisch die Geschwindigkeit. Es entscheidet selbstständig darüber ob es einen grösseren Block, die man dann Windows nennt, oder einen kleineren Block sendet. Jenach Leitungsqualität

Das TCP ist in der [RFC 793][2] definiert.

Die Grafik 15.1 auf der nächsten Seite und die Tabelle 15.1 auf der nächsten Seite zeigen den Aufbau eines TCP-Headers um das wir uns erst einmal kümmern werden.

[1] TCP-Transmission Control Protocol
[2] RFC-793 Transmission Control Protocol

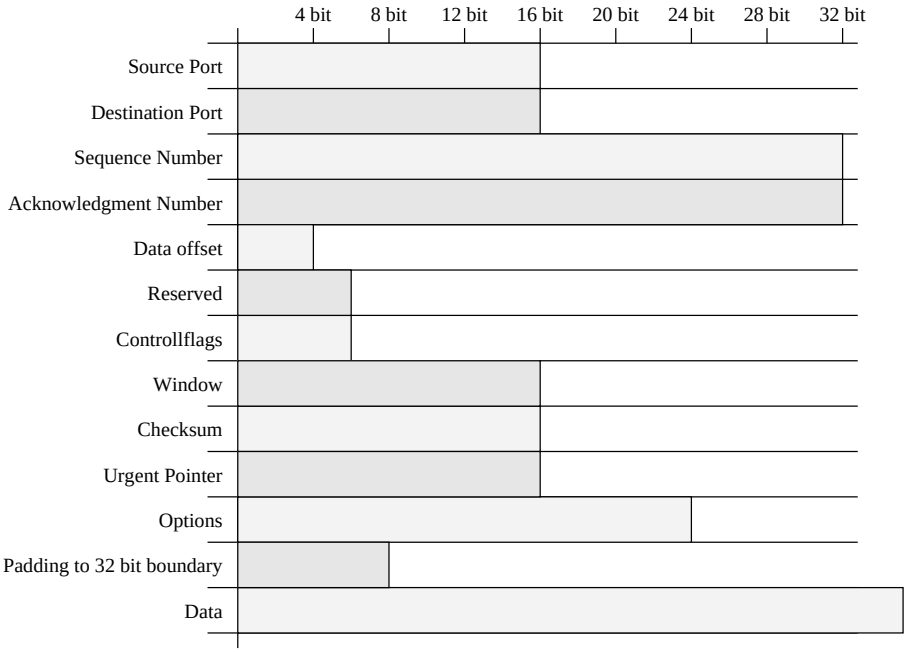

Abbildung 15.1: TCP Header

Tabelle 15.1: Beschreibung des TCP Headers

Feld	**Beschreibung**
Source Port	Nummer des Quellports des Senders der Segmente
Destination Port	Nummer des Zielports des Empfängers
Sequence Number	Jedes Byte wird durchnumeriert. Die Bytereihenfolge wenn man so will. Wenn in den Control Flags das SYN gesetzt ist handelt es sich hier um die ISN[3] und das nächste Segment mit dem ersten Byte bekommt dann ISN+1
Acknowledgment	Gibt die Sequenznummer des letzten erfolgreich übertragenem Segments wieder. Ist das ACK gesetzt beinhaltet das Feld die nächste Sequenznummer die erwartet wird
Data Offset	Bestimmt den Anfang der Daten in 32 bit Schritten. Die Länge des TCP Headers inklusive Optionen sozusagen
Control Flags	Dieses Feld in in der Grafik 15.2 auf der nächsten Seite und in der Tabelle 15.2 auf der nächsten Seite entsprechend erklärt
Window	Die Größe eines Fensters
Checksum	Prüfsumme des TCP Headers
Urgent Pointer	Wird nur beachtet wenn das URG in den Controlflags gesetzt ist. Es gibt die Position von bevorzugten Daten innerhalb eines Fensters an.

▼

[3] ISN-Initial Sequence Number

▲

Options	Das Optionsfield wird in 3 Bereiche eingeteilt. Die ersten 8 bits geben den Typ an, die nächsten 8 bit die Länge der Option und die nächsten 16 bits Daten in anhängigkeit der ersten 8 bits. Siehe Abschnitt 15.1 auf der nächsten Seite
Padding	Auffüller des Headers auf eine 32 bit Grenze. Immer mit NULL'en

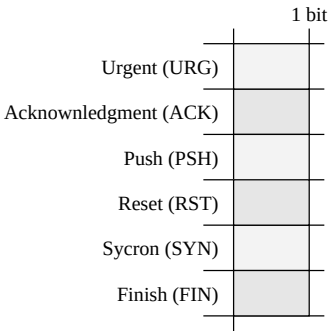

Abbildung 15.2: TCP Controlflags

Tabelle 15.2: Beschreibung der TCP Controlflags

Feld	Beschreibung
URG	Urgent Pointer Feld ist von Bedeutung
ACK	Acknowledgment Feld ist von Bedeutung
PSH	Push Funktion, sorgt für Weiterschaltung von SEQ
RST	Reset der Leitung
SYN	Gleichsetzung der Sequenznummern
FIN	Ende der Übertragung

15.1 TCP Options

Die TCP Options dienen zur Zeit nur um die MSS[4] auszuhandeln. Die ersten 8 bits bilden das Option-Kind Byte. Eine NULL im Option-Kind signaliesiert das Ende der Optionen, eine eins bedeutet keine Option, eine 2 signalisiert die Option MSS. Nur wenn das Option-Kind 2 (MSS) ist bekommen die nächsten 8 bits Bedeutung, diese beinhalten die Länge des Optionsfeld (immer 4 !), die nächsten 16 bits beinhalten die MSS.

Die Option MSS ist nur erlaubt wenn das SYN Flag gesetzt ist. Die MSS kann wie folgt berechnet werden : $MSS \leq MTU - 40$

[4] MSS-Maximum Segment Size

Kapitel 16

FUNKTIONSWEISE VON TCP

Um TCP und die Inhalte der einzelnen Felder im TCP Header zu verstehen werden wir uns nun dem Verbindungsaufbau / Abbau, also die Funktionsweise näher betrachten. Zu jeder einzelnen Verbindung müssen die Addressen der Hosts sowie die Portaddressen bekannt sein. Eine Zieladdresse mit der Portaddresse nennt man **socket**.

16.1 Verbindungsaufbau

Damit man über TCP überhaupt Daten austauschen kann benötigt TCP ersteinmal die Gewissheit das am anderen Ende überhaupt der Host empfangsbereit ist. Zur Realiesierung benutzt TCP dabei den **3-Way Handshake**. Das 3-Wege-Handgeschüttle findet jedesmal statt, wenn TCP eine Verbundung aufbauen will. Wenn der Server nicht antwortet können auch keine Datenübertragungen stattfinden. Die folgenden Punkte beschreiben den Verbindungsaufbau bzw. den 3 Way Handshake. Die Grafik 16.1 verdeutlicht dieses.

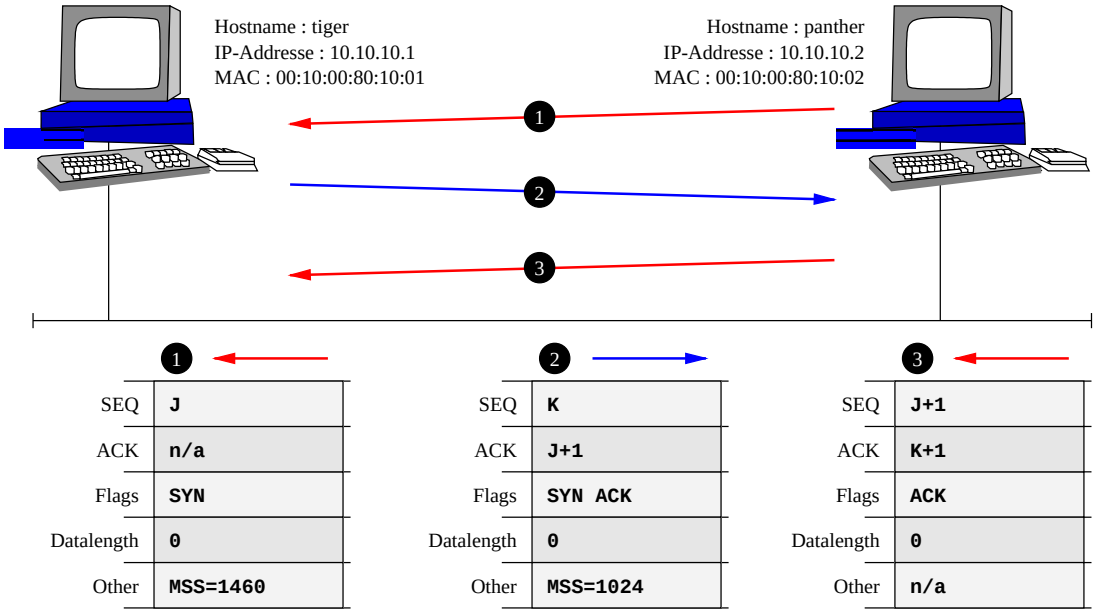

Abbildung 16.1: Der 3-Wege Handshake

1. Der Client, hier PANTHER, sendet ein inhaltsloses Segment an den Server, hier TIGER. Mit diesem Segment kündigt der Client seine Sendewilligkeit an. Dafür denkt sich PANTHER irgendeine *Sequence Number (SEQ) J* aus. Diese *SEQ* muß er für jedes Byte und für jedes Statusbit, bis auf `ACK`, hochzählen, damit der Server weiß an welchen Ort im Datenstrom er das Segment einzuordnen hat. Durch das setzen von `SYN` wird die Sequenznummer automatisch zur ISN[1]

[1] `ISN`-Initialize Sequence Number

2. Der Server, hier TIGER, bekommt nun die Anforderung und prüft ob der *Destination Port* von einem Prozeß belegt ist. Wenn nicht siehe Abschnitt 16.2. Wenn ja, dann speichert der Server die ISN J ab und sendet dem Client eine Bestätigung. Die Acknowlegmentnumber entspricht der ISN des Clients plus 1 durch das SYN $J + 1$. Weiterhin sendet er die gültige *MSS (Maximum Segmentsize)*, und seine eigene ISN K zu.

3. Der Client empängt jetzt nun eine Bestätigung von seiner Sendewilligkeit, ACK. Da der Server selbst Sendewilligkeit durch sein SYN vermittelt, muß sich der Client die ISN K merken und diese Bestätigen mit $K + 1$. Optional kann der Client bereit mit diesem Segment Daten senden, muß aber nicht.

16.2 Erfolglose Verbindungsaufbau

Sollte auf dem Server ein Dienst sich nicht an den Port gebunden haben, dann kommt es zu einem Fehler. Die Fehlermeldung `connection refused` bedeutet das eine Verbindung zum Server möglich ist jedoch der Dienst auf dem Server nicht aktiv ist. Die Fehlermeldung tritt meist auch dann auf wenn der Server ein Segment mit gesetztem RST sendet. Der Client versucht eine Verbindung zu einem zur Zeit nicht laufenden Server-Dienst herzustellen. z.B. Sie versuchen mittels dem Netscape auf einem Server zu surfen, der jedoch kein Webserver gestartet hat. Die Grafik 16.2 zeigt dieses grafisch.

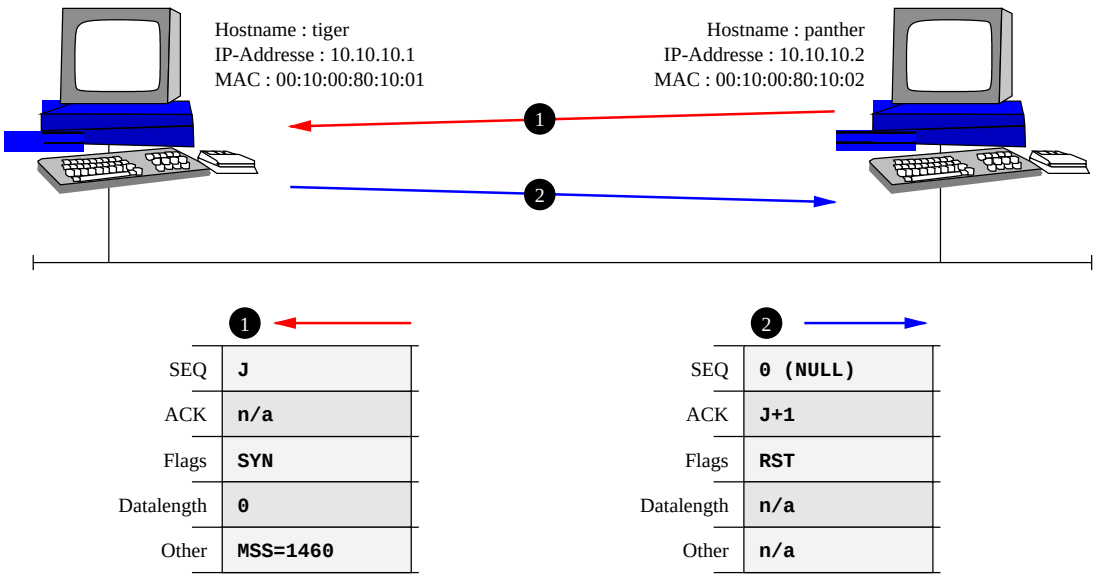

Abbildung 16.2: Ein erfolgloser Verbindungsaufbau

1. siehe Punkt 1 vom Verbindungsaufbau 1 auf der vorherigen Seite

2. Da der Serverprozess nicht läuft sendet der Server das Segment mit einem gesetztem RST Flag zurück. Die Acknowledgmentnummer muß jedoch $J + 1$ sein.

Danach sind diese Verbindungsdaten ungültig.

16.3 Einfache Datenaustausch

Der Datenaustausch funktioniert immer in beiden Richtungen. Wir schaun uns mal den Datenaustausch vom Client zum Server an. Die Grafik 16.3 zeigt auch den anderen Weg, der jedoch genauso aussieht.

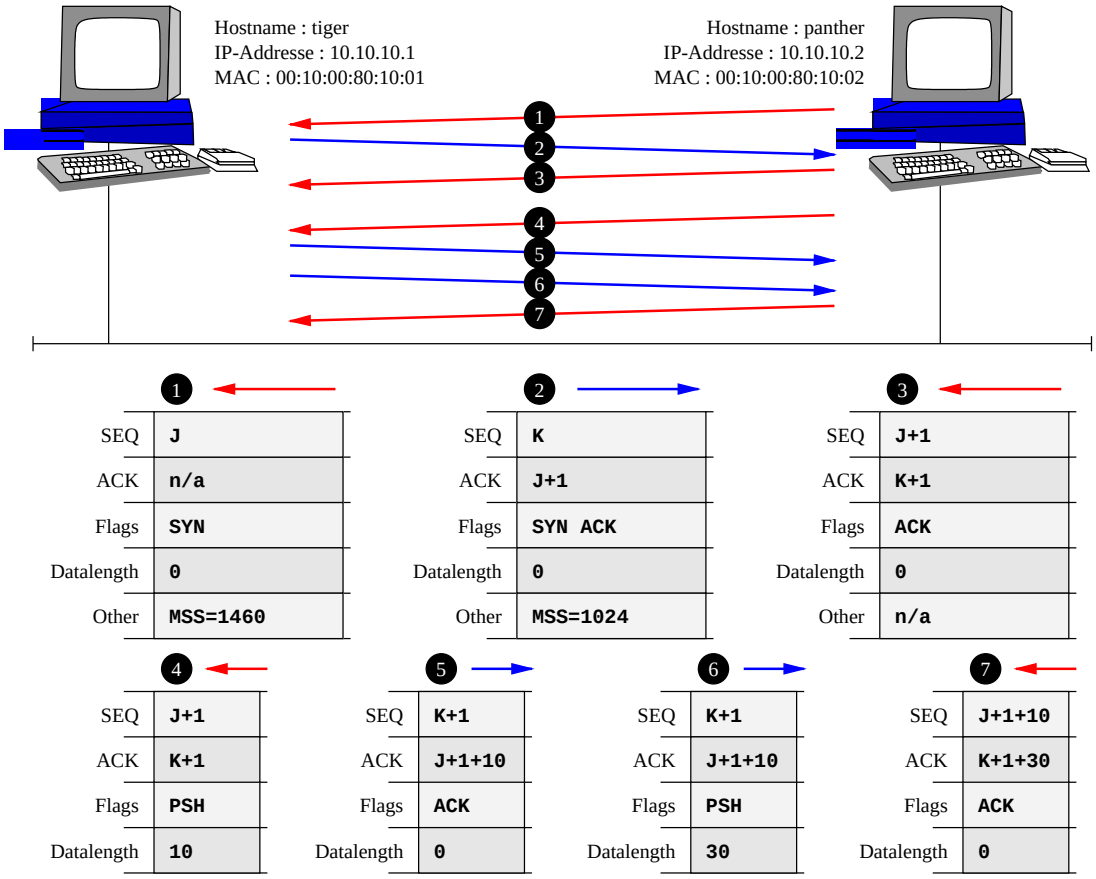

Abbildung 16.3: Einfacher TCP Datenaustausch

1. siehe 3-Wege Handshake

2. siehe 3-Wege Handshake

3. siehe 3-Wege Handshake

4. Nachdem der 3-Wege Handshark abgelaufen ist sendet nun der Client 10 Bytes Daten an den Server. Das kann z.B. irgendeine Anforderung sein. Er benutzt die *SEQ* $J + 1$ weil die Daten die jetzt kommen an die $J + 1$'te stelle im Datenstrom gehören. Weiterhin muß er dem Server signaliesieren das nun Daten mitgesendet werden. Das tut er indem er das PSH setzt.

5. Der Server der nun die 10 Bytes bekommen hat muß das Segment bestätigen. Er verwendet seine eigene *SEQ* $K + 1$ um das Segment zu senden. Bestätigen hat er jetzt alle Bytes die korrekt im Datenstrom angekommen sind, in diesem Fall sind 10 Bytes plus einmal das SYN Flag, er bestätigt nun $J + 1 + 10$

6. Nach der Auswertung der 10 Bytes vom Client sendet er nun das Ergebnis der Auswertung zurück. Er verwendet wieder $K + 1$ als *SEQ*, weil in vorherigen Segment keine Daten gesendet wurden und ACK verändert den zähler nicht. Auch hier muß er das PSH setzen damit der Client weiß das nun Daten kommen.

7. Der Client bekommt nun endlich die Antwort und muß nun die empfangenden Bytes bestätigen. Dazu verwendet er seine eigene *SEQ* $J + 1 + 10$ und bestätigt mit *ACK* $K + 1 + 30$

16.4 Komplexer Datenaustausch

Der Unterschied zwischen einfachen und komplexen Datenaustauschaktionen besteht in der Benutzung von Fenstern, die man Window nennt. Beim einfachen Datenaustausch wird jedes Segment bestätigt, das ist jedoch nicht tragbar und verlangsamt den Datentaustausch erheblich.

Muß einer der beiden Hosts mehr als *MSS*, sprich mehrere Segmente, senden, dann sendet der Host diese Segmente hintereinander weg ohne auf ein Acknowledgment der einzelnen Segmente zu warten. Wieviel der Host am Stück versenden darf wird bei jedem Segmentaustausch im Feld *Window* mitgeteilt. Das *Window* eines Host beinhaltet immer die maximale Fenstergröße dessen was der Host zu verarbeiten mag.

Hat der Sender ein komplettes Window gesendet wartet er auf eine Bestätigung des Empfängers. Das Acknownledgment Feld der Besätigung beinhaltet das zuletzt angekommene Byte, also genau das gleiche wie beim einzelnen Segment.

Jetzt ist es jedoch so daß, wenn die Anwendung des Empfänger den TCP Buffer komplett gelesen hat versendet der Empfänger ungeachtet der Window Grenzen ein Acknowledgement. Wenn der Sendet nun dieses unverhoffte Acknowledgment bekommt berechnet er seine Windowgrenzen entsprechend neu. Wobei hier der Empfänger die Entscheidungsgewalt der Windowgröße besitzt, der Sender muß sich nach richten, er darf drunterliegen aber nicht drüeber.

Die Grafik 16.4 auf der nächsten Seite zeigt den einfachsten der komplexen Datenaustauschsszenarien.

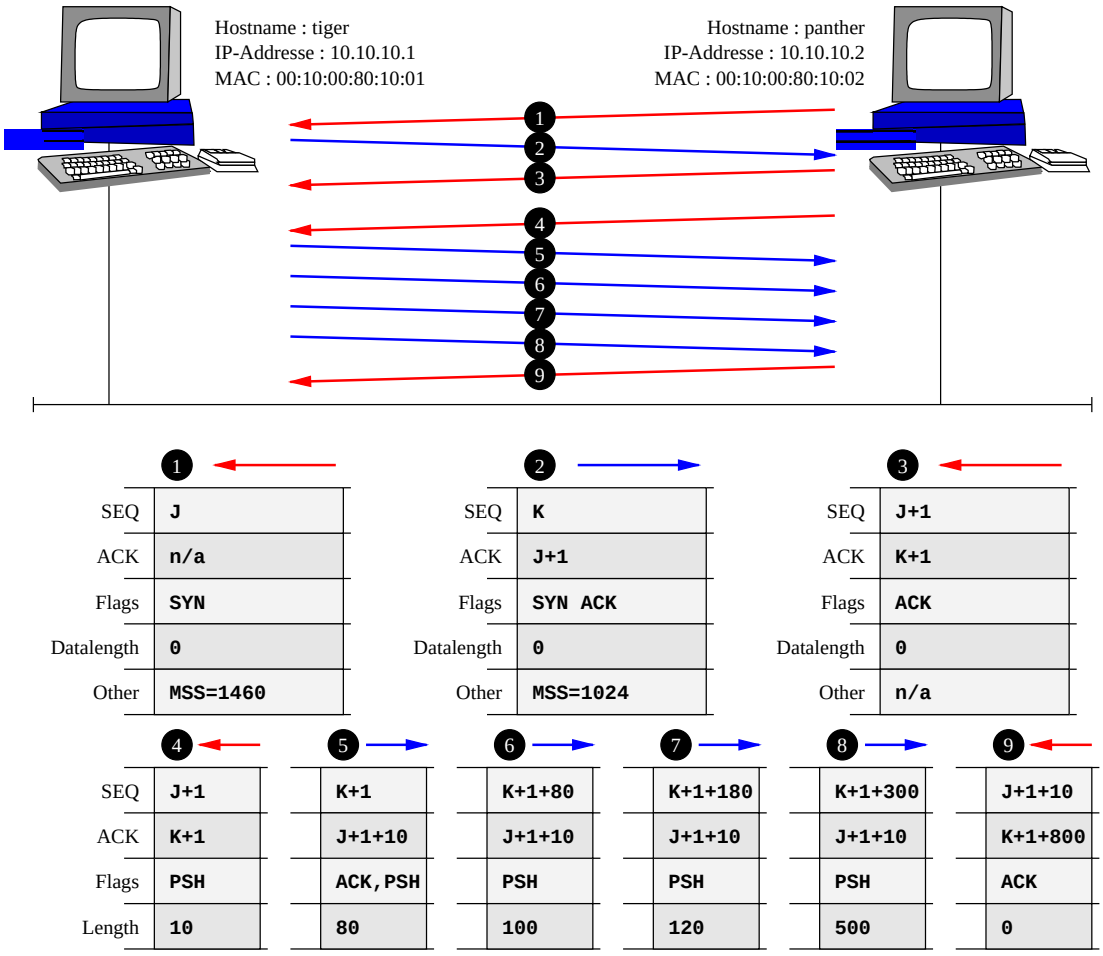

Abbildung 16.4: Komplexer TCP Datenaustausch (einfache Version)

1.

2.

3. siehe 3-Wege Handshake

4. Hier sendet der Client wieder seine Anforderung zum Server

5.

6.

7.

8. Der Server senden dem Client nun das Ergebis zu. Da das Ergebnis mehrere Segmente lanist, sendet er diese ersteinmal ohne auf das ACK vom Client zu warten.

9. Der Client bestätigt nun das zuletzt korrekt gelesende Byte im Datenstrom. $K + 1 + 800$

16.5 Automatische Fehlerbehebung

Das TCP kann automatisch Fehler beheben. Zum Beispiel wenn durch einen Leitungsdefekt einige Segmente verloren gehen, oder aehnliches. Die Anwendung bekommt von all dem nichts mit, das macht TCP vollautomatisch.

Die Lösung des Problems ist recht simpel. Sollte der Sender nicht innerhalb einer gewissen Zeitspanne die Bestätigung, das ACK vom Empfänger, bekommen sendert er einfach die Segmente noch einmal für die er keine Bestätigung hat. Das Acknowledgment vom Empfänger dient dabei als Grenze der empfangenden Bytes.

Jetzt mal angenommen der Empfänger hat bereits ein ACK gesendet, das ist jedoch noch unterwegs in den weiten des Netzes. Dann würde der Empfänger plötzlich zweimal die gleichen Segmente bekommen. Der Empfänger kennt seine eigene aktuelle Acknownlegment Nummer und vergleicht diese mit den Nummer der gerade eingetroffenen Segente. Die sind kleiner als die aktuelle. Der Empfänger verwirft die Segemnte und sendet einfach ein ACK. Am Sender kommen jetzt plötzlich zwei ACK's an. Das kümmert ihn eigendlich überhaupt nicht und verwirft die ACK.

Die Daten sind aber auf jeden Fall angekommen.

16.6 Beenden einer Verbindung

Da unter TCP eine Verbindung mittels 3-way Handshake aufgebaut werden muss, muss auch der Verbindungsabbau entsprechend geplant werden. Ich nenne den Verbindungsabbau die 4-way Termination. Die Grafik 16.5 zeigt den Verbindungsabbau.

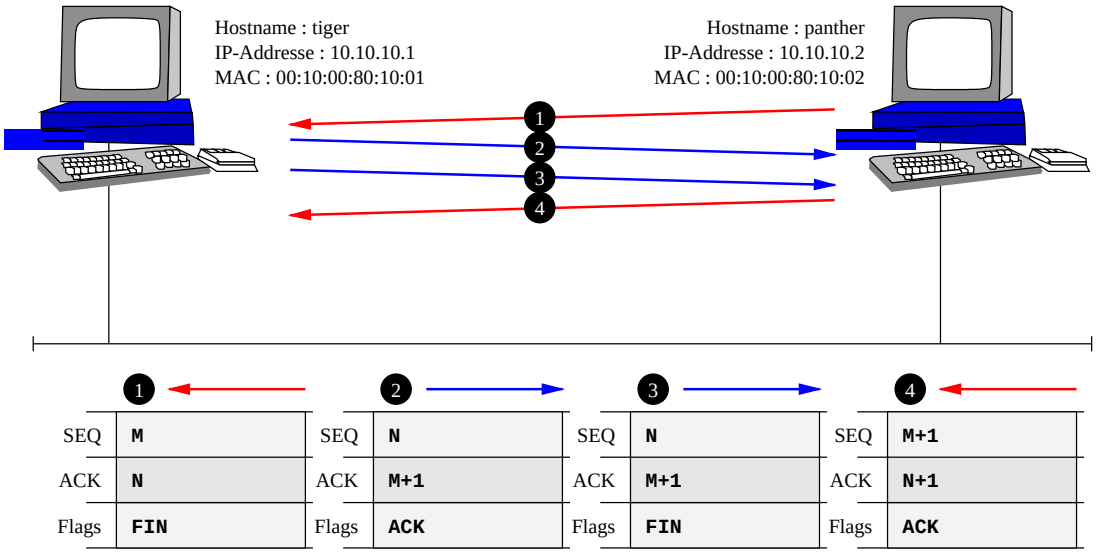

Abbildung 16.5: Die 4 Wege Terminierung

1. Die Anwendung auf dem Client beendet die Verbindung. Diese Anforderung wird mittels gesetztem `FIN` und der *SEQ* M dem Server mitgeteilt. Die *SEQ* M hängt davon ab was vorher über die Leitung lief.

2. Der Server gibt das schliessen der Verbindung an die Anwendung weiter. Der Server sendet dem Client danach ein Acknowledgment mit der $M + 1$. Er selbst vervendet die *SEQ* N Der Client

bekommt sein ACK und wartet nun auf das FIN des Servers

3. Die Anwendung auf dem Server beendet nun die Verbindung. Der Server sendet dem Client ein FIN.

4. Der Client muß das FIN bestätigen

16.7 Ein kompletter Datenaustausch

Die ganzen einzelnen Vorgänge können auch komprimiert werden so das es bei einer simplen und einfachen Übertragung zu extrem wenigen Wegen kommen kann. Die Grafik 16.6 zeigt dieses.

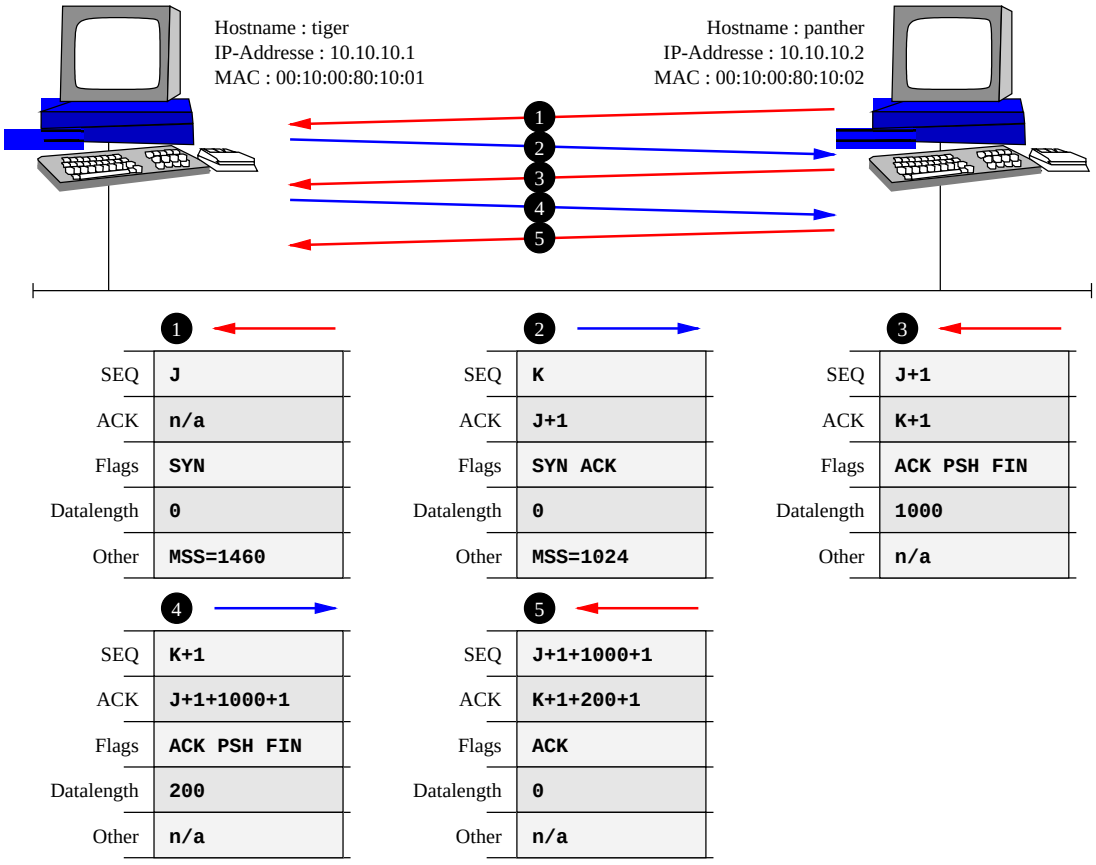

Abbildung 16.6: Ein kompletter Datenaustausch

TEIL IV

ROUTING

- Grundsätzliches
  - IP Addressierung
  - Die Zusammenhänge
  - Übungsaufgaben
- Grundlagen des Routings
  - Router
  - Routing Table
  - Beispiel Netzwerk
- Routing
  - Locales Routing
  - Netzrouting
  - Default Route
  - Dem Routing Mechanismus auf’s Bit geschaut

Kapitel 17

GRUNDSÄTZLICHES

Bevor wir jedoch zum Routen kommen noch einige wichtige Nebensächlichkeiten, wie die Addressierung, die wir auch noch benötigen.

17.1 IP Addressierung

Wir wissen daß ein Host eine IP-Addresse benötigt. Wir wissen auch das eine IP-Addresse aus 32 bits besteht. Aber nicht nur Hosts benötigen eine Addressen, es gibt einige Addressen mit Sonderbedeutungen.

17.1.1 Netzwerk Addressen

Eine Netzwerkaddresse ist eine IP Addresse für einen physikalischen Kabelstrang. Jedes Kabel hat eine IP Addresse, diese IP Addresse nennt man Netzwerkaddresse, oder kurz Netzwerk. Die Addresse ist vergleichbar mit der IPX[1]-Strangnummer von NOVELL. Um jetzt eine Netzwerkaddresse von Hostaddressen zu trennen gibt es die

17.1.2 Netzmasken

die auch Netmask, oder Subnetmask genannt werden. Es handelt sich bei der Maske um einen 32 bit breiten Wert der linken Seite mit einser und auf der rechten Seite mit nuller gefüllt ist. Sieht man sich die kleine Grafik 17.1 an sieht man eine Maske links gefüllt mit einsern rechts gefüllt mit nullen. Dardurch ergibt sich eine Grenze, die Trennung der Maske. Durch die Trennung entstehen zwei Seiten. Die linke Seite ist der **Netzwerkanteil** und die rechte Seite ist der **Hostanteil**.

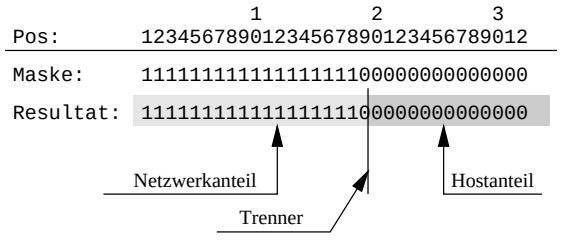

Abbildung 17.1: Maskenbeispiel

Die Grafik 17.2 auf der nächsten Seite zeigt die Verwendung mit einer IP Addresse, und deren Sonderbedeutungen.

[1] IPX-Internet Packet Exchange

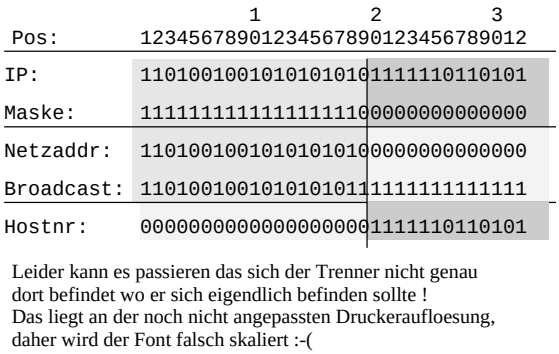

Abbildung 17.2: Maskenbeispiel mit Broadcast, Netzaddresse

Netzwerkaddresse Die Netzwerkaddresse ergibt sich aus der Übernahme des Netzwerkanteils und durch das löschen des Hostanteils mit Nullen. Das Resultat ist die Netzwerkaddresse. Man kann die Netzwerkaddresse auch berechnen :

$$Netzwerkaddresse = IP \textbf{ AND } Maske$$

Broadcastaddresse Die Broadcastaddresse ergibt sich aus der Übernahme des Netzwerkanteils und durch das löschen des Hostanteils mit einser. Das Resultat ist die Broadcastaddresse. Auch diese Addresse kann berechnet werden :

$$Broadcastaddresse = (IP \textbf{ AND } Maske) \textbf{ OR NOT } Maske$$

Hostnummer Die Hostnummer ergibt sich aus dem löschen des Netzwerkanteils und der übernahme des Hostanteils. Auch die Hostnummer kann berechnet werden :

$$Hostnummer = IP \textbf{ AND NOT } Maske$$

Die Maske kann vom Netzwerk Admininistrator selbst bestimmt werden. Unter welchen umständen der Administrator die Maske verändert wird noch geklärt. Es gibt jedoch Standardeinstellungen an die man sich halten kann oder nicht. Sie gelten nur im Internet als 100%-ig wichtig, nicht jedoch im eigenen Netzwerk.

17.1.3 Internet Standardmasken

Im Internet gibt es einen Standard der einige Masken und Addressbereiche definiert. Die Tabelle 17.1 zeigt die Übersicht.

Klasse	ersten 8 Bit der IP-Addr	Bereich von - bis	Verwendete Maske	Masken bits
A	$0xxxxxxx_2$	0 - 127	$11111111000000000000000000000000_2$	8
B	$10xxxxxx_2$	128 - 191	$11111111111111110000000000000000_2$	16
C	$110xxxxx_2$	192 - 223	$11111111111111111111111100000000_2$	24

Tabelle 17.1: Internet Standard IP Klassen

17.1.4 Broadcastaddressen

Wie eine Broadcastaddresse berechnet wird hatt wir ja schon, doch was hat die Broadcastaddresse für eine Aufgabe ? Well, mit einer Netzwerkaddresse spricht man das Netzwerk an, mit der Broadcastaddresse spricht man alle Rechner innerhalb eines Netzwerkes an. Ein Rundruf sozusagen.

17.1.5 Hostnummer

Die Hostnummer gibt die relative Nummer des Host innerhalb eines Netzwerkes an.

17.2 Die Zusammenhänge

Jedes Kabel hat eine Addresse. Jeder Host der an diesem Kabel angeklemmt ist, sollte den gleichen Netzwerkanteil wie das Kabel aufweisen. Das bedeutet das die Anzahl der anklemmbaren Host innerhalb eines Netzwerkes begrenzt ist. Man kann zur Berechnung der maximalen Hosts an einem Netzwerk die folgende Formel verwenden :

$$Anzahl = 2^{AnzahlBits} - 2$$

Die minus 2 steht für einmal die Netzwerkaddresse, da ist der Hostanteil auf Null, und die Broadcastaddresse, dort ist der Hostanteil auf Eins. Diese Formel ist auch auf die Anzahl der Netzwerke anwendbar. Folglich kommt man zur folgendem Ergebnis das in der Tabelle 17.2 festgehalten wurde :

Class	Netzwerk bits	Anzahl Netzwerke	Host bits	Anzahl Hosts		Max Hosts dieser Class
A	7	126	24	16.777.214		2.113.928.964
B	14	16.382	16	65.534		1.073.577.988
C	21	2.097.150	8	254		532.676.100
				Summe		3.720.183.052

Tabelle 17.2: Anzahl der Internet Standard Netzwerke und Hosts

Wie man der Tabelle entnehmen kann fehlen noch 574.784.244 mögliche Hostaddressen, diese halbe Million geht fuer Netzwerkaddressen und Broadcastaddressen drauf.

Aber wieder zurück zu unseren Hostaddressen. Um nun alle Angaben zu ermitteln benötigen wir **immer** eine **Maske**. Für unser Beispiel definieren wir mal ein 22'er Maske : `11111111111111111111110000000000`. Und wir sagen mal wir haben es mit einer X-beliebigen IP-Addresse tu tun : `10101010100010101111011011001010`. Zur besseren Übersicht dient wohl die Tabelle 17.3 auf der nächsten Seite. Als erstes stellt man mit den Regeln um eine Netzwerkaddresse zu erhalten die Netzwerkaddesse fest. Der Host liegt also im Netzwerk `10101010100010101111010000000000`, somit kann fast der Standpunkt des Hosts bestimmt werden. Durch die Netzwerkaddresse erhalten wir auch die Broadcastaddresse `10101010100010101111011111111111`. Die Hostnummer des Host steht immer relativ zum Netzwerk, d.h. in jedem Netzwerk gibt es einen Host 1, die Hostnummer des Hosts hier lautet `00000000000000000000001011001010`, was man jedoch aus Einfachheit einfach vom Binären ins Dezimale umrechnet. Der Host hat also die Nummer : $2^1 + 2^3 + 2^6 + 2^7 + 2^9 = 714$. Es ist der 714'te Host in diesem Netzwerk.

Bedeutung	Wert
Maske	11111111111111111111110000000000
IP Addresse	10101010100010101111011001010
Netzwerkaddresse	10101010100010101111010000000000
Broadcastaddresse	10101010100010101111011111111111
Hostnummer	00000000000000000000001011001010

Tabelle 17.3: Beispiel einer Netzwerkaddressberechnung

17.3 Übungsaufgaben

Die Tabelle 17.4 auf der nächsten Seite ist entsprechend auszufüllen. Berechnen Sie die Werte der leeren Zellen.

IP-Addresse	Maskenbits	Maske	Netzwerkaddresse	Broadcastaddresse	Hostnummer
10.65.1.1	16	255.255.0.0	10.65.0.0	10.65.255.255	257
210.127.23.5	24				
154.165.54.84		255.255.255.0			
40.50.60.85	28				
205.54.23.40		255.255.240.0			
			11.12.13.64	11.12.13.127	12
	18		34.45.128.0		542

Tabelle 17.4: Übungsaufgaben zur Berechnung von Addressen

Kapitel 18

GRUNDLAGEN DES ROUTINGS

Eine Route ist eine Wegbeschreibung zu einem Host bzw. zu einem Netzwerk.

18.1 Router

Da die Anzahl der Host pro physikalisches Kabel begrenzt ist benötigt man mehrere physikalische Kabel. Um die einzelnen physikalischen Stränge miteinander zu verbinden, um die Dienste eines Hosts auf einem anderen physikalischen Strang zu benutzen, müssen die einzelnen Stränge mittels **Router** verbunden werden.

Ein paar Stichpunkte. Ein Router

hat immer mehr als 2 Netzwerkkarten, von daher wird er auch meist **Multi-Home-Host** genannt, weil er in mehreren Netzen zu Hause ist

leitet Datenpakete in andere Netze aus

kann auf Wunsch auch Pakete Filtern, um eine gewisse Sicherheit im Lokalen Netzwerk zu erreichen

ist in den meisten Fällen kein eigener Rechner sondern ein flaches stück Kasten mit dem Erscheinungsbild eines Hubs

leitet keine Broadcasts in andere Netze weiter, das wäre ja auch Wahnsinn

kennt **immer** nur den nächsten Router am angeschlossenen lokalen Netzwerk

Um dem Router jetzt zu sagen wo er welches Netzwerk findet benötigt man die

18.2 Routing Table

Eine Routing Table beinhaltet für jeden Eintrag folgende Informationen :

1. Zielnetzwerk in Form von einer Netzwerkaddresse. Auch Destination DN genannt
2. Die Gateway[1] (auch GW genannt) Addresse wo das Netzwerk zu finden ist. **Das Gateway muß sich immer im lokalem Netzwerk befinden**
3. Die Maske des Zielnetzwerkes. Auch NM genannt
4. Die Anzahl der Hops[2]
5. Typ des Eintrages. Entweder Locale Route, Gateway Route oder Auto Route
6. Das Netzwerk-Device. Genannt das Interface.

[1] `Gateway`-Allgemeines Wort für einen Multiprotocol Router
[2] `Hops`-Anzahl der dazwischen liegenden Router

Jeder Host benötigt **immer** eine Routingtable.

Ein solcher Eintrag kann z.B. so umschrieben werden : Das Netzwerk DN mit der Maske NM findest du am Gateway GW. Dieses Gateway (Router) weis dann wie es weiter geht.

18.3 Beispiel Netzwerk

Wir wollen uns ab jetzt nun auf das Beispielnetzwerk, siehe Grafik 18.1 konzentrieren.

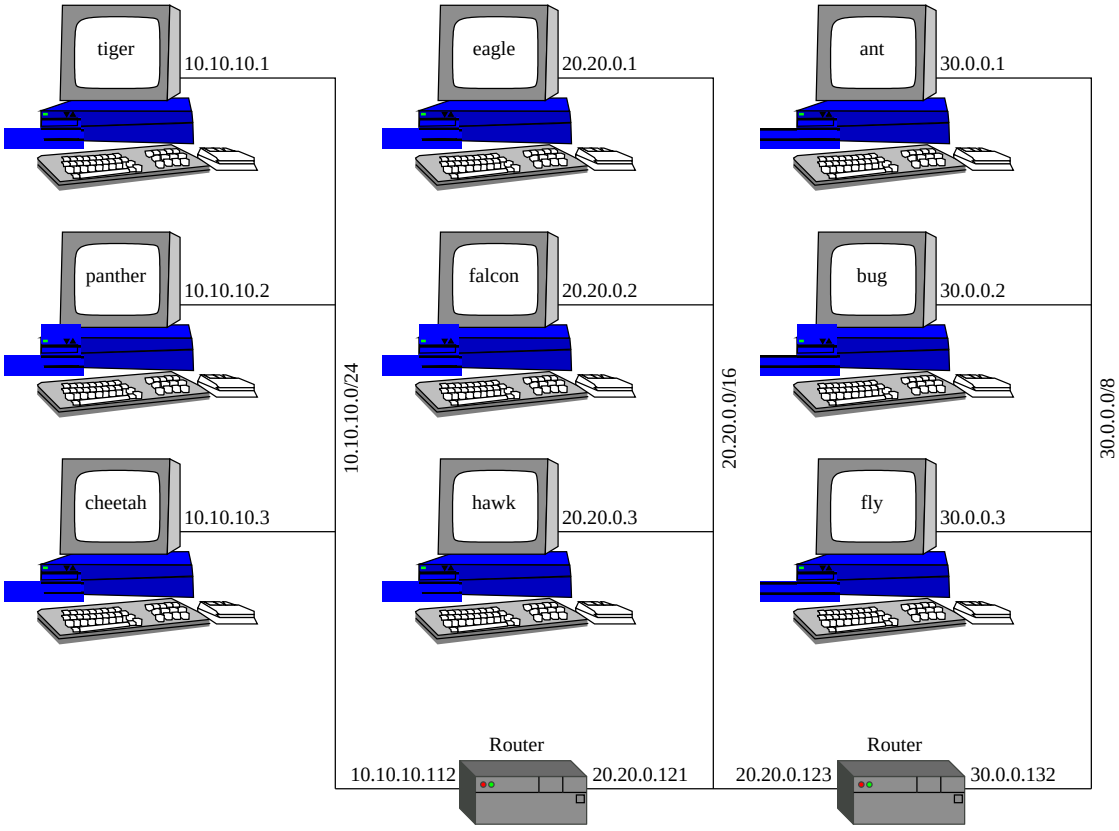

Abbildung 18.1: Routing Beispiel 01

Kapitel 19

ROUTING

Jetzt gehts los ...

19.1 Locales Routing

Eine Locale Netzroute ist nur bei sehr flexiblen Netzwerkbetriebssystemen notwendig. Nur UNIX ist in der Lage die Netzkartenkonfiguration von der localen Route zu trennen. In den meisten Fällen benötigt man aber keine Trennung. Dieser Abschnitt ist also nur für richtige Betriebssysteme wichtig.

Wir betrachten uns mal den Host TIGER . Dieser Host hat die IP-Addresse `10.10.10.1/24`. Damit der Host nun auch mit PANTHER komunizieren kann benötigt TIGER die Angabe einer Route in das Locale Netzwerk. Als Gatewayaddresse kommt hier seine eigene Schnittstellenaddresse zum Einsatz, das Device kann auch dafür genommen werden. Es ist jedoch besser die IP-Addresse anzugeben um nur einmal eine Devicebindung, in der Konfiguration, zu erreichen. Das bedeutet das der erste Routingeintrag aus der Tabelle 19.1 zu entnehmen.

Tabelle 19.1: Localrouting Tabelle für TIGER

Destination	Gateway	Netmask	Metric
`10.10.10.0`	`10.10.10.1`	`255.255.255.0`	0

Somit kann TIGER nun auf alle Hosts im Localen Netzwerk ein Ping absetzen. Sofern auf PANTHER eine Localeroute gesetzt wurde. Der Router RT1 benötigt zwei locale Routen, weil er in zwei Netzwerken zu Hause ist. Die Routingtabelle 19.2 verdeutlicht dieses.

Tabelle 19.2: Localrouting Tabelle für RT1

Destination	Gateway	Netmask	Metric
`10.10.10.0`	`10.10.10.112`	`255.255.255.0`	0
`20.20.0.0`	`20.20.0.121`	`255.255.0.0`	0

Der Host FALCON benötigt logischerweise auch eine Locale Route, wie die Routingtable 19.3 zeigt.

Tabelle 19.3: Localrouting Tabelle für FALCON

Destination	Gateway	Netmask	Metric
`20.20.0.0`	`20.20.0.2`	`255.255.0.0`	0

Der Router RT2 befindet sich wie RT1 auch in zwei Netzen, somit ergibt sich die Routingtable 19.4 auf der nächsten Seite

Tabelle 19.4: Localrouting Tabelle für RT2

Destination	Gateway	Netmask	Metric
20.20.0.0	20.20.0.123	255.255.0.0	0
30.0.0.0	30.0.0.132	255.0.0.0	0

Der Host ANT bekommt natürlich auch eine Routingtable 19.5

Tabelle 19.5: Localrouting Tabelle für ANT

Destination	Gateway	Netmask	Metric
30.0.0.0	30.0.0.1	255.0.0.0	0

So nachdem alle Hosts sich im localen Netzwerken unterhalten können wenden wir uns mal den Routing in andere Netzwerke zu ...

19.2 Netzrouting

Dem Host TIGER muß bekannt gegeben werden wie er in das 20'er Netz kommt. Zum Beispiel will TIGER eine Verbindung zum FALCON aufbauen. TIGER muß also wissen wohin er seine Daten senden soll, weil FALCON sich nicht im lokalen Netzwerk befindet. Damit TIGER in das FALCON Netz gelangt muß er über den RT1 gehen. Der RT1 verbindet die beiden Netze miteinander, er betätigt sich also als Router bzw. als Gateway. D.h. wir müssen TIGER sagen daß er das 20'er Netzwerk am Gateway RT1 findet. Die Routingtable 19.6 verdeutlicht dieses.

Tabelle 19.6: Netzrouting Tabelle für TIGER

Destination	Gateway	Netmask	Metric
10.10.10.0	10.10.10.1	255.255.255.0	0
20.20.0.0	10.10.10.112	255.255.0.0	1

Wir haben gesagt das Gateway ist immer local und es weiss wie es weiter gehen soll. Und das trifft zu, da der RT1 in beiden Netzen zuhause ist kennt er das 20'er Netz und leitet das Paket entsprechend weiter. Wenn nun FALCON das Paket bekommt, will er meist dem TIGER antworten. Um das zu tun braucht FALCON eine Route zum TIGER bzw. in das 10'er Netz. Woher sollte auch FALCON wissen das er das 10'er Netzwerk über das Gateway RT1 erreichen kann ?!? Also benötigt FALCON die entsprechenden Einträge in der Routingtable 19.7

Tabelle 19.7: Netzrouting Tabelle für FALCON

Destination	Gateway	Netmask	Metric
20.20.0.0	20.20.0.2	255.255.0.0	0
10.10.10.0	20.20.0.121	255.255.255.0	1

Damit ANT auch mit FALCON Daten austauschen kann, benötigt ANT eine Route in das 20'er Netz-

werk. Wie TIGER auch. Der RT2 kenn auch wieder das 20'er Netzwerk und somit ist das der Ansprechpartner von ANT . Der ensprechende Eintrag zeigt die Routingtable 19.8

Tabelle 19.8: Netzrouting Tabelle für ANT

Destination	Gateway	Netmask	Metric
30.0.0.0	30.0.0.1	255.0.0.0	0
20.20.0.0	30.0.0.132	255.255.0.0	1

Wenn FALCON allerding Antworten möchte benötigt er die Route ins 30'er Netzwerk. FALCON bekommt einen weiteren Eintrag in seine Routingtable 19.9

Tabelle 19.9: Netzrouting Tabelle für FALCON

Destination	Gateway	Netmask	Metric
20.20.0.0	20.20.0.2	255.255.0.0	0
10.10.10.0	20.20.0.121	255.255.255.0	1
30.0.0.0	20.20.0.123	255.0.0.0	1

FALCON ist nun der einzige Host der mit allen Host im Netz sich verbinden kann. TIGER kann zwar mit FALCON eine Verbindung aufnehmen jedoch noch nicht mit ANT . Wir wollen aber das die beiden sich unterhalten können. Wir müssen TIGER sagen wo er das 30'er Netzwerk findet. Und das 30'er Netzwerk findet er am Gateway RT1 und **NICHT** am Gateway RT2 , die Addresse von RT2 kann der Admin des 10'er Netzes nicht wissen. TIGER soll also seine Daten für das 30'er Netzwerk zum RT1 senden. Die Routingtable 19.10 zeigt den Eintrag.

Tabelle 19.10: Netzrouting Tabelle für TIGER

Destination	Gateway	Netmask	Metric
10.10.10.0	10.10.10.1	255.255.255.0	0
20.20.0.0	10.10.10.112	255.255.0.0	1
30.0.0.0	10.10.10.112	255.0.0.0	2

Die Daten jedoch bleiben am RT1 ersteinmal hängen weil RT1 nicht weiss wo er das 30'er Netzwerk finden kann, es ist ja nicht local. Also benötigt RT1 einen weiteren Eintrag in seine Routingtable. Die Tabelle 19.11 zeigt den Eintrag. Das Gateway ist der RT2 er kennt das 30'er Netzwerk und leitet es entsprechend weiter.

Tabelle 19.11: Netzrouting Tabelle für RT1

Destination	Gateway	Netmask	Metric
10.10.10.0	10.10.10.112	255.255.255.0	0
20.20.0.0	20.20.0.121	255.255.0.0	0
30.0.0.0	20.20.0.123	255.0.0.0	1

Jetzt kommen die Daten bei ANT an, aber ANT will Antworten und versucht nun mit dem 10'er

Netzwerk eine Verbindung zu erstellen, nur zu dumm das ANT das 10'er Netzwerk nicht kennt. Also helfen wir Ihm etwas in dem wir einen weiteren Eintrag in seine Routingtable schreiben. Die Tabelle 19.12 zeigt den Eintrag.

Tabelle 19.12: Netzrouting Tabelle für ANT

Destination	Gateway	Netmask	Metric
30.0.0.0	30.0.0.1	255.0.0.0	0
20.20.0.0	30.0.0.132	255.255.0.0	1
10.10.10.0	30.0.0.132	255.255.255.0	2

Der RT2 legt die Daten jedoch nach /dev/null ab weil er das 10'er Netz nicht kennt. Also geben wir Ihm das 10'er Netzwerk bekannt. Die Tabelle 19.13 zeigt den Eintrag.

Tabelle 19.13: Netzrouting Tabelle für RT2

Destination	Gateway	Netmask	Metric
20.20.0.0	20.20.0.123	255.255.0.0	0
30.0.0.0	30.0.0.132	255.0.0.0	0
10.10.10.0	20.20.0.121	255.255.255.0	1

Der RT1 kennt das 10'er Netz und leitet die Daten weiter.
Es ist geschafft.

19.3 Default Route

In einem grossen Netzwerk kommt so eine sehr grosse Anzahl von Routingeinträgen zusammen. Um das etwas zu vermindern gibt es den Mechanismus der Default Route. Eine Default Route ist nichts anderes als ein Routing Eintrag mit dem Zielnetzwerk 0.0.0.0 und einer Maske von 0.0.0.0, das trift auf alle möglichen Kombinationen zu. Der Defaulteintrag wird immer dann genommen wenn die Routingtabelle komplett durchsucht wurde. Der Host sendet dann dieses Paket zu das Gateway, mit dem Gedanken : Soll er sich doch drum kümmern. Betrachtet man den Host TIGER dann muß er das 20'er Netzwerk und das 30'er Netzwerk über RT1 senden. Dem TIGER könnte man jetzt die Default Route auf dieses Gateway setzen. Danach sendet TIGER ALLES mit dem er nichts anfangen kann nach RT1 . Besonderst im Internet hilfreich, das spart ca. 16 Millionen Einträge. Die neue Routingtabelle von TIGER zeigt uns die Tabelle 19.14

Tabelle 19.14: Netzrouting Tabelle für TIGER

Destination	Gateway	Netmask	Metric
10.10.10.0	10.10.10.1	255.255.255.0	0
0.0.0.0	10.10.10.112	0.0.0.0	1

Somit sendet TIGER alles was er nicht lokal senden kann an das Gateway. Das Gateway weiss dann wie es weiter gehen soll, hoffendlich.
include redirectroute

19.4 Dem Routing Mechanismus auf’s Bit geschaut

Dieser Abschnitt soll den Vorgang im innern des Rechners beim Routing verdeutlichen. Die Grafik 19.1 auf der nächsten Seite zeigt den Ablauf des Routings.

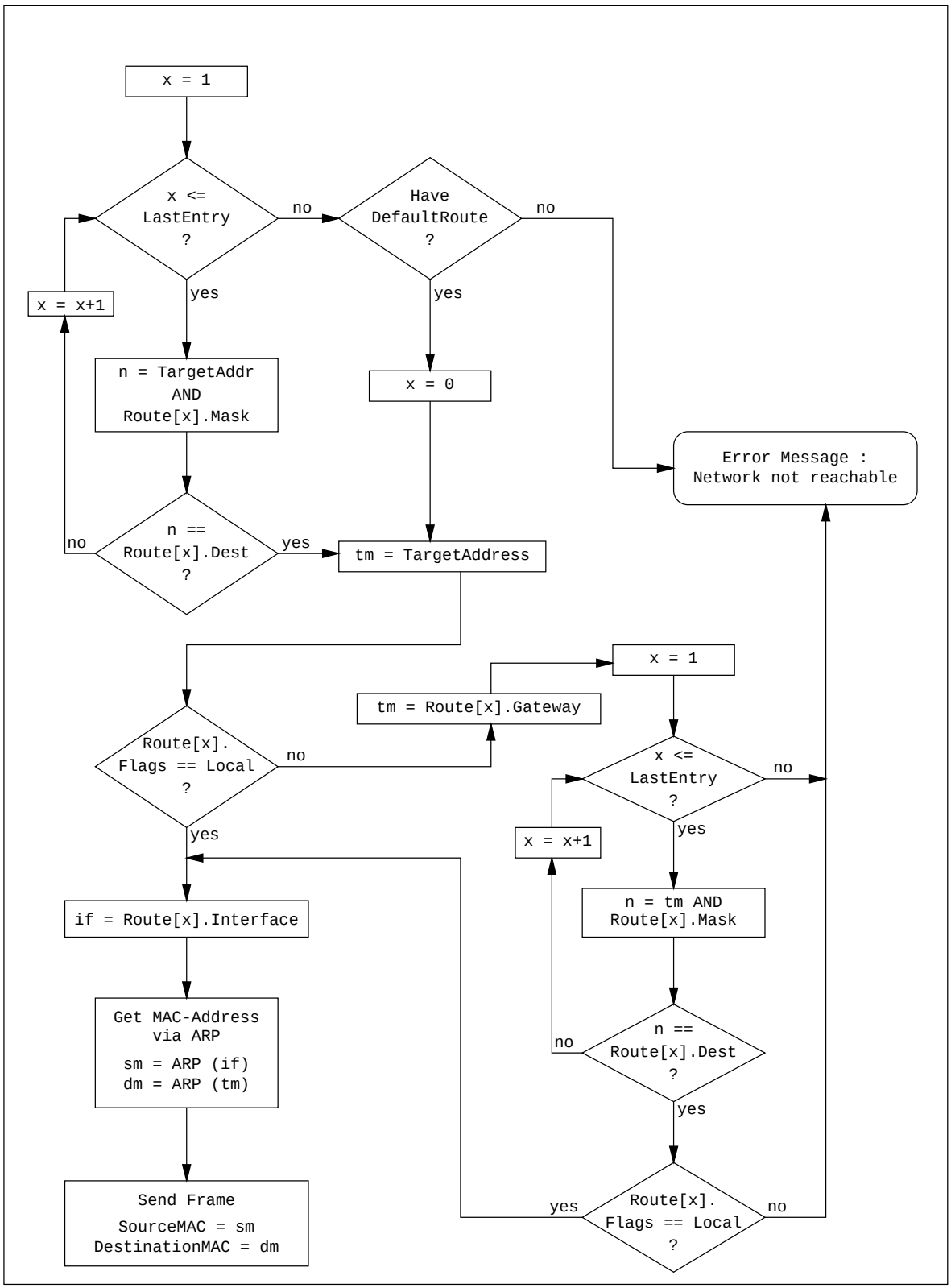

Abbildung 19.1: Programmablaufplan beim Routing

TEIL V

IP CONFIGURATION

- Einführung
- SuSE
- Linux Allgemein
  - Kernelmodule
  - Interface Konfiguration
  - Routing Konfiguration
  - Troubleshooting
- Praktikum
  - Ein einfaches Netzwerkwerk
  - Virtuelle Netze
  - Netz mit Routern

Kapitel 20

EINFÜHRUNG

Grundsätzlich gilt : Wenn ein OS installiert wird, wird man meist durch die IP-Configuration durchgeführt. Dieser Teil kümmert sich um die Konfiguration bei einem bereits installierten System.

Kapitel 21

SuSE

Bei der SuSE gestaltet sich die Configuration der IP-Addresse recht simpel. Unter der SuSE Distribution gibt es ein Administrations Tools mit dem Namen `Yast` der fast alles kann.

Bevor man jedoch die Konfiguration der Netzwerkkarte vornimmt sollte man vorher in den Runlevel `S` wechseln um die Netzwerkkonfiguration komplett zu entlanden.

```
tiger:root # init S
Shutting down .......
....
Give root password to login:
tiger: root #
```

Bildschirmausschnitt 21.0.1: Wechseln in Runlevel S unter SuSE

Danach ruft man den `Yast` auf. Im Menüpunkt *Administration des Systems* befinden sich alle nötigen Funktionalitäten zur Einstellung der Netzwerkkarte und anderes auch. Bei der SuSE 6.1 muß zuerst die Netzwerkkarte eingetragen werden. Im Menupunkt *Hardware in System integrieren* befindet sich die Funktionalität *Netzwerkkarte konfigurieren.* Nach der Wahl macht sich ein Dialog **??** auf Seite ?? auf. Dort muß die entsprechende Hardware eingetragen werden.

Der *Typ des Netzwerks* ist bei Ethernetkarten immer `eth` wobei die nachfolgende Ziffer die Nummer der Netzwerkkarte angibt. Die *Art der Netzwerk-Karte* bestimmt den zuverwendenen Treiber. Mit *Optionen zum Laden des Moduls* können noch Optionen zum Laden des Moduls mit gereicht werden. Siehe SuSE Handbuch.

Nachdem die Konfiguration abgeschlossen ist muß noch die IP Addresse konfiguriert werden.

Wieder im Yast Menubaum angelagt findet man unter *Administration des Systems* den Punkt *Netzwerk konfigurieren* dort macht sich ein weiteres Menu auf. Der Menupunkt *Netzwerk Grundkonfiguration* öffnet wieder ein Dialog **??** auf Seite ??.

Die Spalte Nummer hat nichts mit der Device-Nummer zutun. Als allererstes muß das *Device* festgelegt werden. Zu wählen ist dort dann Ethernet **??** auf Seite ??.

Danach muß die IP-Addresse festgelegt werden. Geben Sie dort die entsprechenden Werte ein **??** auf Seite ??.

Nachdem nun alles Konfiguriert wurde muß der `Yast` verlassen werden. Was der `Yast` nicht kann sind Routingeinträge zu anderen Netzen. Unter SuSE wird die `/etc/route.conf` ausgewertet. Diese Datei muß mit einem Editor entsprechend angepasst werden, hier gilt allerdings : Lesen der `man route.conf` ! Ein Eintrag kann ca. so aussehen :

Quelltext 21.0.1 Beispiel eines Routingeintrags in der `/etc/route.conf` unter SuSE Linux

```
# destination gateway        netmask        device
#-------------------------------------------------
10.65.13.0    0.0.0.0        255.255.255.0  eth0   # local route
10.65.200.0   10.65.13.200   255.255.255.0  eth0   # net route
```

Auf der Kommandozeile wechseln Sie wieder in den Runlevel 2.

```
tiger:root # init 2
Shutting down .......
Starting ............

tiger login:
```

Bildschirmausschnitt 21.0.2: Wechseln in Runlevel 2 unter SuSE

Wichtig ist es nun das Systemlog /var/log/messages zu prufen !

Kapitel 22

LINUX ALLGEMEIN

Dieser Abschnitt beschäftigt sich mit der Konfiguration mit einem Linux Kernel ab 2.0.3x

22.1 Kernelmodule

Damit Linux überhaupt mit irgendwelchen IP Konfigurationen belästigt werden kann benötigt Linux den Treiber für die Netzwerkkarte. Normalerweise ist ein solches Modul direkt in den Kernel eingebunden um höchste Performance zu erreichen. Zum probieren eignet sich jedoch der Mechanismus der Modulladens unter Linux. Das Programm

```
insmod [options] [ -o aliasname ] modulname [ symbol=value ]*
```

ermöglicht das laden von Modulen. Um zum Beispiel die Intel Ethernet Express Pro 100 als modul zuladen benötigt man den Namen des Moduls. Alle Module liegen unter /lib/modules. Das Modul heisst eepro100. Mit dem Befehl `insmod eepro100` wird das Modul für die Netzwerkkarte geladen. Kommt es zu einem Fehler bekommen Sie bescheid, ansonsten schweigt das Programm. Bei einer NE2000 ist vorher das Modul 8390 zu laden, es definiert eine Schnittstelle zur den Karten.

Wenn mehrere gleiche Netzwerkkarten in einem Rechner sich befinden (siehe Router) muss für jede Karte ein Modul geladen werden. Es ergibt sich dann jedoch ein Problem das ein Modul nur einmal mit gleichen Namen geladen werden kann. Dazu benötigt man die *-o* Option mit der man den Modul einen anderen Namen geben kann. Und es sollte auf jeden Fall geklärt werden welche Karte man mit welchen Modul ansprechen möchte. Das Betriebssystem geht einfach hin und verteilt immer den nächstfreien Deviceeintrag dem nächsten Modul was kommt. d.H. um zu verhindern das in einem Router das eth0 plötzlich auf der unteren Karte landet nur wiel die obere Karte nicht schnell genug geantwortet hat, gibt man entweder die Portnummer oder die IRQ Nummer als Option=Value dem Modul mit auf den weg. Siehe Beispielscript 22.1.1 auf der nächsten Seite.

Quelltext 22.1.1 Beispiel eines Hardwarekonfigurationsscripts

```
#!/bin/sh
#
# Dieses Script Konfiguriert den rt1 mit zwei NE2000 Netzwerkkarten
# Karte 1. Port = 0x300 IRQ = 10 (obere)
# Karte 2. Port = 0x340 IRQ = 11 (untere)
#
# Zuerst das gemeinsame Schnittstellenmodul der NE2000 laden
insmod 8390 || exit 1

# Jetzt die beiden Karten laden
insmod -o ne0 ne port=0x300 || exit 1
insmod -o ne1 ne port=0x340 || exit 1

exit 0
```

Zum Entladen der Module kann

```
rmmod modulname
```

verwendet werden. Das Auflisten der bereits geladenen Module übernimmt

```
lsmod
```

22.2 Interface Konfiguration

Um einem Interface eine IP Addresse zu vergeben muß das entsprechende Modul geladen sein, und es darf nicht Benutzt werden. Das Programm

```
ifconfig [interface]
ifconfig interface aftyp [option]+
```

dient zur Konfiguration und zur Anzeige der Konfiguration. Als *aftyp* - Address Families kommen die in der Tabelle 22.1 aufgelisteten Typen in Frage.

Tabelle 22.1: Address Families von ifconfig

Option	Beschreibung
inet	TCP/IP
ax25	AMPR Packet Radio
ddp	Appletalk Phase 2
ipx	Novell IPX
netrom	AMPR Packet radio

Eine Aufstellung der *option*en zeigt Tabelle 22.2 auf der nächsten Seite. Es gibt einige Optionen die erwarten eine weiter Angabe w.z.B. broadcast erwartet weiterhin die Broadcastaddresse. Die Optionen up und down z.B. erwarten keine weiteren Optionen.

Tabelle 22.2: Optionen von ifconfig

Option	Beschreibung
`up`	Aktivierung des Interfaces
`down`	Deaktivierung des Interface
`[-]arp`	[De]Aktiviert das ARP auf diesem Interface
`mtu N`	Setzt die MTU für dieses Device
`netmask NM`	Setzt die Netzmaske
`broadcast BR`	Setzt die Broadcastaddresse
`address`	Setzt die Addresse des Interfaces
`multicast`	Setzt die Multicasteigenschaften
`[-]allmulti`	[De]Aktiviert den promiscuous mode. Soll bedeuten das alle Frames hoch in den NAL und zum Kernel gesendet werden. Erlaubt Monitoring

Die Reihenfolge scheint bei Linux immer noch ein Manko zu sein. Zuerst kommt die Addresse danach die Netmask und dann erst die Broadcast. Um jetzt den RT1 weiterhin zu installieren kann man das Script 22.2.1 verwenden.

Quelltext 22.2.1 Beispiel eines IP-Konfigurationsscripts

```
#!/bin/sh
#
# Dieses Script konfiguriert die IP Addressen von rt1
# Voraussetzung : Die Module sind geladen und die obere ist eth0
#

# eth0 Konfigurieren. Die geht in das 10'er Netzwerk
ifconfig eth0 inet 10.10.10.112 netmask 255.255.255.0 broadcast 10.10.10.255

# eth1 geht in das 20'er Netzwerk
ifconfig eth1 inet 20.20.0.121 netmask 255.255.0.0 broadcast 20.20.255.255

exit 0
```

22.3 Routing Konfiguration

Routen können mit dem Programm

```
route
route add [-net|-host] Destination [netmask MN] [gw GW] [metric me-
tric]
route del [-net|-host] Destination [gw GW] [netmask MN] [metric me-
tric]
```

erstellt, gelöscht und angezeigt werden. Auch hier ist die Reihenfolge ernst zunehmen. Weiterhin ist UNIX, und NOVELL in der Lage eine Hostroute zu erstellen. Es kommt mit einer IP Addresse als Ziel und einer 32'er Maske gleich. Eine Hostroute kann sich sehr stark auf die Sicherheit im positiven Sinne auswirken. Gut, um den RT1 jetzt nun noch die Routingkonfiguration mitzugeben kann man den Quelltext 22.3.1 benutzen.

Quelltext 22.3.1 Beispiel eines Routing Konfigurationsscripts

```
#!/bin/sh
#
# Dieses Script konfiguriert das Routing von rt1
#

# Routen in die beiden angeschlossenen Netze
route add -net 10.10.10.0 netmask 255.255.255.0 gw 10.10.10.112
route add -net 20.20.0.0 netmask 255.255.0.0 gw 20.20.0.121

# Route in das 30'er Netz
route add -net 30.0.0.0 netmask 255.0.0.0 gw 20.20.0.123
exit 0
```

22.4 Troubleshooting

22.4.1 arp -a

22.4.2 traceroute ziel-ip

22.4.3 netstat

22.4.4 ping

Kapitel 23

PRAKTIKUM

Dieses Kapitel soll helfen das Routing und die IP Addressvergabe entsprechend zu üben. Bevor es jedoch ernst wird hier einige Voraussetzungen :

1. Nach der Installation ändern Sie bitte das Startupmodul /etc/rc.d/network und fügen Sie nach dem `start)` ein `insmod eepro100` ein. Damit das Modul beim Hochfahren geladen wird. Nach `stop)` fügen Sie ein `rmmod eepro100` ein so das dieses Modul beim verlassen wieder rausgeworfen wird. Das Modul sollte wie folgt aussehen, siehe Quelltext 23.0.1

Quelltext 23.0.1 Startupmodul für die Netzwerkkarte

```
#!/bin/sh
#
# Modulscript /etc/rc.d/network
#
case "$1" in
        start)
                insmod eepro100
                ;;
        stop)
                ifconfig eth0 down
                rmmod eepro100
                ;;
        *)
                echo "usage $0 start|stop"
                exit 1
esac
exit 0
```

2. Das Startmodul /etc/rc.d/route welches die Routen aus /etc/route.conf liest und konfiguriert, kann gelöscht werden. Vergessen Sie nicht die Links in den Runlevelverzeichnissen mit zu löschen.

3. Booten Sie nachdem das System

4. Danach hat Ihr Host keine IP mehr !

Für jedes hier zu erstellende Netzwerk ist ein Script zu schreiben, daß im Verzeichnis /root/bin abzulegen ist. Das Script hat die Aufgabe bei Aufruf das Netzwerk für diesen Rechner zu installieren. Der Name des Scripts wird entweder bei jeder Übungsaufgabe bekanntgegeben oder Sie denken sich einen Namen aus.

Bei dem Aufbau der Netzwerke sollten Sie nach dem 7 Punkte Plan vorgehen

1. Wählen Sie einen Masteradmin, der dann in Erscheinung tritt, wenn sich zwei oder mehr Gruppen die Unternetze betreuen streiten. Ausserdem koordiniert der Mastervisor die Netzwerkenstehung. Der Masteradmin verteilt Masken, Netzaddressen, etc.

2. Berechnen Sie nun die Anzahl der Netzwerkstänge und bilden Sie passene Gruppen für jedes Netzwerk. Jede Gruppe hat einen Groupadmin, der die Gruppe mit IP Addressen sowie als Vermittler zwischen den anderen Gruppen und dem Masteradmin fungiert

3. Die Groupadmins und der Masteradmin verkabeln nun jedes Netzwerk physikalisch mit einander

4. Die einzelnen Arbeitsstationen sind so zu konfigurieren das jeder Host alle Host im localen Netzwerk anpingen kann

5. Die Groupadmins verkabeln und konfigurieren die Router für die jeweiligen Localen Netze. Erst wenn der Router in den localen Netzwerken vollständig verfügbar ist gehts weiter

6. Die Router und alle anderen Hosts werden nun für die restlichen Netzwerke konfiguriert

7. Jeder Host muss alle anderen Hosts an'`ping`'en können

23.1 Ein einfaches Netzwerkwerk

Ziel dieser Übung ist es die Verbindung zu allen Rechnern über ein physikalisches Kabel herzustellen. Benötigt werden eine entsprechende Anzahl von Hubs um alle Host miteinander zu verbinden. bei 16'er Hubs genügen in der Regel zwei Stück.

1. Verbinden Sie alle Rechner, mit außnahme der Router und des Druckers, mit den Hubs. So das es ein physikalisches Netzwerk wird. Vom Dozentenrechner wird ein Anschluss zur Verfügung gestellt

2. Das Script soll `net-10.65.0.0` heissen

3. Benennen Sie einen Administrator der für die Verteilung der IP Addressen verantwortlich ist

4. Konfigurieren Sie nun alle Host für das Netzwerk. Benutzen Sie die Netzwerkaddresse `10.65.0.0/16`. Die IP Addresse der Hosts bekommen Sie vom Administrator Die `10.65.1.1` ist bereits Reserviert.

5. Erstellen Sie den nötigen Routingeintrag

6. Stellen Sie kontakt mittels `ping` zu den anderen Hosts her.

7. Helfen Sie den anderen beim Konfigurieren

23.2 Virtuelle Netze

Ziel dieser Übung ist es zu verdeutischen das ein oder mehrere Host an einem physikalischen Strang nicht unbedingt auch von der anderen erkannt werden. Fangen Sie erst mit der Übung an wenn alle mit der anderen Übung fertig sind.

1. Benennen Sie 4 Administratoren. Die Administratoren bilden jeweils eine gleich grosse Gruppe von Teilnehmern/Hosts
2. Die 4 Administratoren denken sich jeweils eine Class C Addresse aus und klären das mit den anderen Administratoren. Jede Gruppe soll eine eigenständige Addresse besitzen
3. Jeder Administrator verteilt nun die zur Verfügung stehenden IP-Addressen aus dem Pool an die Gruppenteilnehmer
4. Konfigurieren Sie nun alle Hosts mit den verteilten IP-Addressen. Das Script soll `virtual` heißen
5. Konfigurieren Sie Ihre einene Localroute
6. Obwohl alle Host an einem physikalischen Strang hängen ist es Ihnen nicht möglich die anderen Hosts der anderen Gruppen zu erreichen. Überlegen Sie wie Sie dem abhelfen können und korrigieren Sie das Problem. Ohne die Router zu verwenden

23.3 Netz mit Routern

Ziel ist es nun das Netzwerk mit Routern aufzubauen. Die vorherige Übung muß jedoch vollständig abgearbeitet sein. Der Scriptname soll `net1` sein.

1. Tennen Sie nun die einzelnen Gruppen, indem Sie für jede Gruppe/Netzwerk ein eigenen Hub verwenden.
2. Überprüfen Sie nach der Verkabelung unbedingt die ping-fähigkeit der einzelnen Hosts in den Netzwerken. Löschen Sie die Netzrouten der anderen Netze wieder.
3. Stellen Sie nun die Router auf. Verkabeln Sie zuerst nur ein Netz mit dem Router.
4. Konfigurieren Sie den Router. Er muß in dem angeschlossenen Netzwerk vollständig ping-fähig sein.
5. Verbinden Sie nun das 2. Netzwerk mit dem Router und konfigurieren Sie diesen auch für das Netzwerk.
6. Nun werden in beiden Netzwerken die Routen in das andere Netz konfiguriert. Die Ping-fähigkeit ist danach zu prüfen.
7. Nachdem dieses abgeschlossen ist können nun die Routen in die unliegenen Netze konfiguriert werden. Abgeschlossen ist das ganze wenn jeder jeden Pingen kann.

TEIL VI

SUBNETZE

- Grundlagen des Subnetting
  - Beispiel einer Vernetzung
  - Zweite Beispiel einer Vernetzung mit Subnetting
- Total Subnetting
  - Grundlagen
  - Workstation W1
  - Workstation W2
  - Router R1
  - Router R2
  - Router R3
  - Router R4
  - Router R5
  - Router R6
  - Zusammenfassung

Kapitel 24

GRUNDLAGEN DES SUBNETTING

Dieser Teil soll den Begriff Subnetting erklären.

Subnetting wird überall und überall sehr intensiv betrieben. Es dient zur Reduzierung des IP-Addressbereichs den ein physikalischer Netzwerkstrang im Normalfall beanspruchen würde.

24.1 Beispiel einer Vernetzung

24.1.1 Situation

In einem Unternehmen sind Sie als Netzwerkadministrator eingesetzt und bekommen nun von Ihrem Vorgesetzen den Auftrag zwei Büros zu vernetzen. Da die beiden Büros zu weit von einander getrennt sind entschliessen Sie sich dafür einen Router zwischen den beiden Büros zu installieren. Sie vergeben je eine Netzwerkaddresse, Sie entschliessen sich für eine `Class C` Addresse. Ihr Vorgesetzter hält Ihnen wärendessen ein Kabel ins Gesicht und sagt : *Da müssen die Büros dran !*. Gesagt getan. Sie installieren einen weiterern Router der die beiden Büros mit dem unternehmensweiten Intranet verbindet. Siehe dazu die kleine Grafik 24.1.

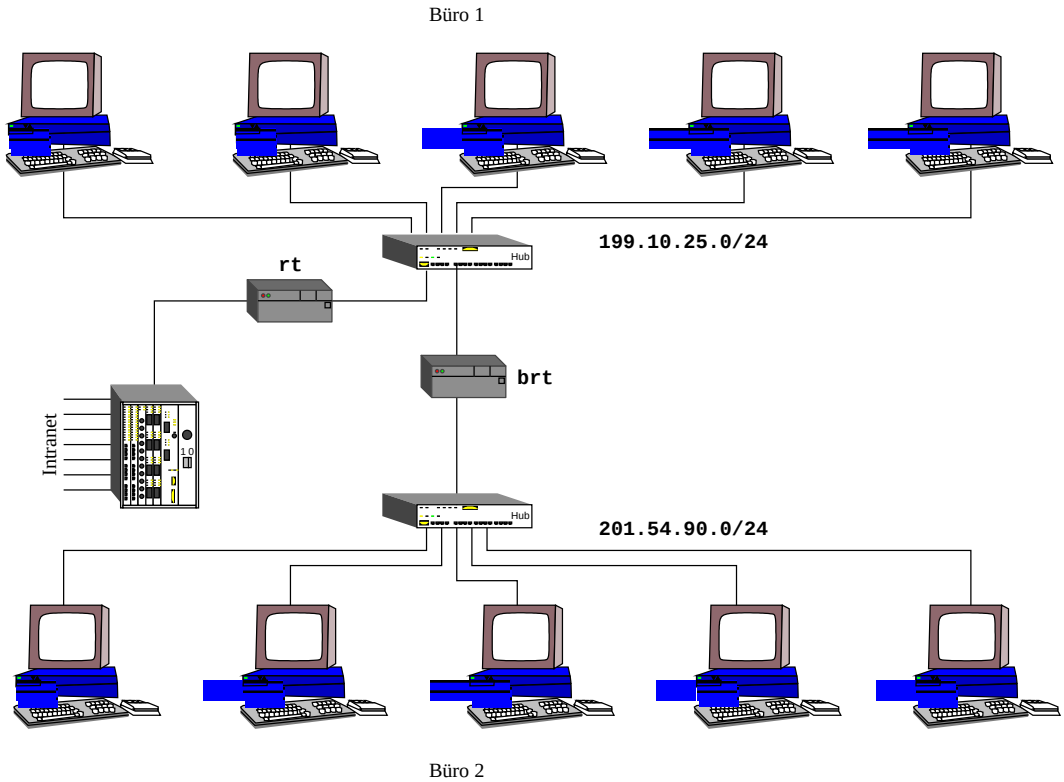

Abbildung 24.1: Beispiel Netzwerk für Büros

Nach dem Aufbau gehen Sie zum Netzwerkadministrator und teilen Ihm die Netzwerkaddressen der beiden Büros mit. Der rennt darauf hin los und passt nun alle Routingtabellen der ganzen Router und Server an. Da er sich darüber freut sagt er Ihnen *Danke !*. Sie sind mit der Arbeit fertig, fahren nach Hause und schlafen. Am nächsten Morgen kommen Sie wieder in das Unternehemen und der Netzwerkadministrator rennt immer noch rum ...

24.1.2 Analyse

Zur Analyse sammeln wir erst einmal die `positiven` Ereignisse :

1. Sie selbst hatten die wenigste Arbeit

So das wars, sammeln wir nun einmal die `negativen` Ereignisse :

1. Sämtliche Routingtabellen der Router müssen entsprechend angepasst werden, es kann dadurch eventuell zu Ausfällen kommen, z.B. durch Tipfehler
2. Sämtliche Server die die Arbeitsstationen im Netzwerk verwenden müssen, müssen auch entsprechend angepasst werden. Auch hier kann es eventuell zu Ausfällen kommen
3. Wird im Intranet der Dienst des Internet's benutzt, in Form von E-Mail, WWW, News, Ftp, etc. muß auch eine eventuell konfigurierte Firewall entsprechend angepasst werden
4. Hängt das Intranet direkt am Internet mit festen Addressen, dann müssen alle Router im Internet entsprechend angepasst werden, sind ja nur ein paar zehntausend
5. Ein zentaler Router bekommt zwei zusätzliche Routingeinträge die bei jedem Paketdurchlauf mitgeprüft werden müssen, bei weiteren solchen Konstelationen verringert sich der Durchsatz eines solchen Routers
6. Dem Unternehmen gehen durch diese beiden Büros satte 512 IP-Addressen durch die Lappen, obwohl vielleicht nur 20 effektiv genutzt werden
7. Wenn Sie noch ein Büro vernetzen, hat der Netzwerkadministrator wieder Arbeit

Ich höre jetzt auf ...

24.1.3 Lösung

Nun man muß nun versuchen ohne grossen Aufwand die beiden Büros zu vernetzen. Die Lösung dazu lautet Subnetting. Wie wir bereits wissen kennt jeder Router IMMER nur seinen local angeschlossenen Nachbar-Router. Wir wissen auch wie der Routingmechanismus intern Funktioniert. Der Router geht die Routingtabelle durch und rechnet für jedes Paket die mögliche Route aus.

Er nimmt die *Zieladdresse* und *AND'*et diese mit der *Netzmaske* im Tabelleneintrag. Er vergleicht danach das Ergebnis mit dem *Destination Network* aus der Tabelle. Passt der Eintrag, hat der Router sein *Gateway* gefunden.

Mit diesem Wissen geligt es uns das Problem auf einfachste Art und Weise zu lösen.

Zu erst jedoch müssen die beiden Büro Netzstränge eingeschränkt werden, wollen wir nur mal so annehmen daß maximal 20 IP-Addressen in jedem Büro gebraucht werden. Wenn es mehr in absehbarer Zukunft werden sollen muß das berücksichtigt werden. Wie wir nun wissen entscheidet die Netzmaske

über den Hostanteil und Netzwerkanteil einer IP-Addresse. In unserem Beispiel haben wir eine 24'er Maske verwendet.

Frage: Wer sagt das es eine 24'er Maske sein muß ?

Keiner ! Nur mal so rein theoretisch : Was würde passieren wenn wir die Maske einfach etwas nach rechts schieben würden ?

Nun der Hostanteil wird kleiner und der Netzwerkanteil wird größer. Toll.

Theoretisch gesprochen würde es ja reichen wenn man die Maske soweit nach rechts schiebt bis nur die geforderte Anzahl von möglichen Hostnummern in das Bitfeld passt, oder ?

Nun, wenn wir planen das maximal 20 Arbeitsstationen plus maximal 2 Router sowie die Broadcastaddresse und die Netzwerkaddresse, dann kommen wir auf maximal 24 IP-Addressen im Hostbereich. Rechnet man die 24 ins Binäre um bekommt man 11000_2. 5 Bits würden reichen um die 24 IP-Addressen darzustellen. Was jedoch nicht möglich ist ist eine Trennung auf eine spezielle Zahl sondern nur auf die Anzahl von Bits. Bei 5 Bits bekommt man jedoch $2^5 = 32$ mögliche IP-Addressen. Nun das kann man nicht verhindern.

Wir wissen jetzt das wir nur 5 Bits benötigen um ein Netzwerk für ein Büro zu erstellen. Die Maske könnten wir jetzt rein theoretisch zu einer 27'er Maske machen. Somit hätten wir $2^5 - 2 = 30$ mögliche Host IP-Addressen. Passt also rein.

Etwas ist jedoch noch dazu gekommen, wir haben 3 Bits im Netzwerkanteil gewonnen. Der Standard sagt eine 24'er Maske wir haben jedoch eine 27'er Maske. Mit diesen 3 gewonnenen Bits könnte man nun rein theoretisch $2^3 = 8$ Netzwerke erstellen. Jedoch auch hier gilt Broadcast und Netzwerkaddresse ist auszuschliessen. Also nur $2^3 - 2 = 6$ Netzwerke. Da diese Netzwerke jedoch unterhalb des eigendlich echten Netzwerken liegen nennt man diese gewonnenen 6 Netzwerke auch **Subnetze**

Die Frage ist nun : *Was soll das bringen ?* OK ! nehmen wir mal an Sie wenden Subnetting auf Ihr Büromodell an. Büro 1 wäre z.B. das erste Subnetz und Büro 2 das zweite Subnetz. Sie könnten dann noch 4 weitere Büros vernetzen, ohne die Routingtabellen im gesamten Netzwerk zu ändern.

Weil : Der Router RT der mit dem Intranet verbunden ist, den konfigurieren Sie. Da jeder Router immer nur seinen nächsten Router kennt, wissen die Router im Intranet garnicht wie es hinter Ihrem Router RT aussieht ! oder ? Was für die Router gilt kann man auf die einzelnen Administratoren übertragen. Man könnte jetzt sagen *OK ! Sende das Netz xyz/24 an Router rt, der weiß wie es weiter geht !*

Und genau das passiert auch so !

24.2 Zweite Beispiel einer Vernetzung mit Subnetting

24.2.1 Situation

Sie bekommen wieder den Auftrag 2 Büros zu vernetzen. Sie gehen nun als allererstes zu Ihmen vorgesetzen Netzwerkadministrator und fragen Ihn nach einer Netzwerkaddresse die Sie benutzen dürfen. Der Netzwerkadmin gibt Ihnen die `198.12.92.0`. Ihnen ist schnell klar das dieses eine `Class C` mit einer 24'er Maske ist. Weiterhin gibt er Ihnen eine Gateway IP-Addresse aus dem Intranet `199.99.99.92`, die der Router auf der anderen Seite haben muß.

24.2.2 Berechnung

Da Sie zwei Büros vernetzen sollen, brauchen Sie auch zwei Netzwerkaddressen. Jetzt liegt es an Ihnen die Maske entsprechend zu verschieben. Dazu bieten sich zwei lösungen an :

1. Es werden wohl nie mehr als 2 Subnetze (Büros) werden. Der Gedanke die Maske nur 2 Bit's zu verschieben kommt hier in Betracht. $Subnetze = 2^2 - 2 = 2$. Somit könnten Sie $Hosts = 2^6 - 2 = 62$ IP-Addressen in einem Büro vergeben

2. Es könnten mehr Subnetze werden, jedoch ist eins klar : Mehr als 20 Hosts werden es niemals im Büro, weil der Platz vielleicht garnicht dazu ausreicht. Man könnte die Maske nun um 3 Bit's schieben. $Hosts = 2^5 - 2 = 30$ reicht vollkommen aus. Jetzt jedoch haben Sie die Möglichkeit $Subnetze = 2^3 - 2 = 6$ Netzwerke (Büros) zu vernetzen

Diese Entscheidung hängt vom Unternehmen ab, expandiert es stark, oder hängt von der räumlichen Dimension ab, passen 60 Geräte überhaupt dort rein ?

Wir entscheiden uns für 6 Subnetze und 30 Hosts. Wir arbeiten also mit einer 27'er Maske.

Da jedes Betriebssystem die Maske Dezimal erwartet, müssen wir die noch ausrechnen :

```
   Subnetz-Maske : 11111111.11111111.11111111.11100000
         Dezimal :      255.     255.     255.     224
```

Das gleiche gilt für die Netzmasken und Broadcastaddressen :

```
 Netzaddresse : 11000110.00001100.01011100.00000000
                      198.      12.      92.       0

Subnetz-Maske : 11111111.11111111.11111111.11100000
                      255.     255.     255.     224

Subnetz 1 ;
 Netzaddresse : 11000110.00001100.01011100.00100000
                      198.      12.      92.      32

    Broadcast : 10111101.00001100.01011100.00111111
                      198.      12.      92.      63

Subnetz 2 ;
 Netzaddresse : 11000110.00001100.01011100.01000000
                      198.      12.      92.      64

    Broadcast : 11000110.00001100.01011100.01011111
                      198.      12.      92.      95
```

24.2.3 Aufbau

Der Aufbau des Netzes unterscheidet physikalisch nicht dem von vorher, jedoch kommen nun die IP-Addressen dazu. Siehe Grafik 24.2 auf der nächsten Seite

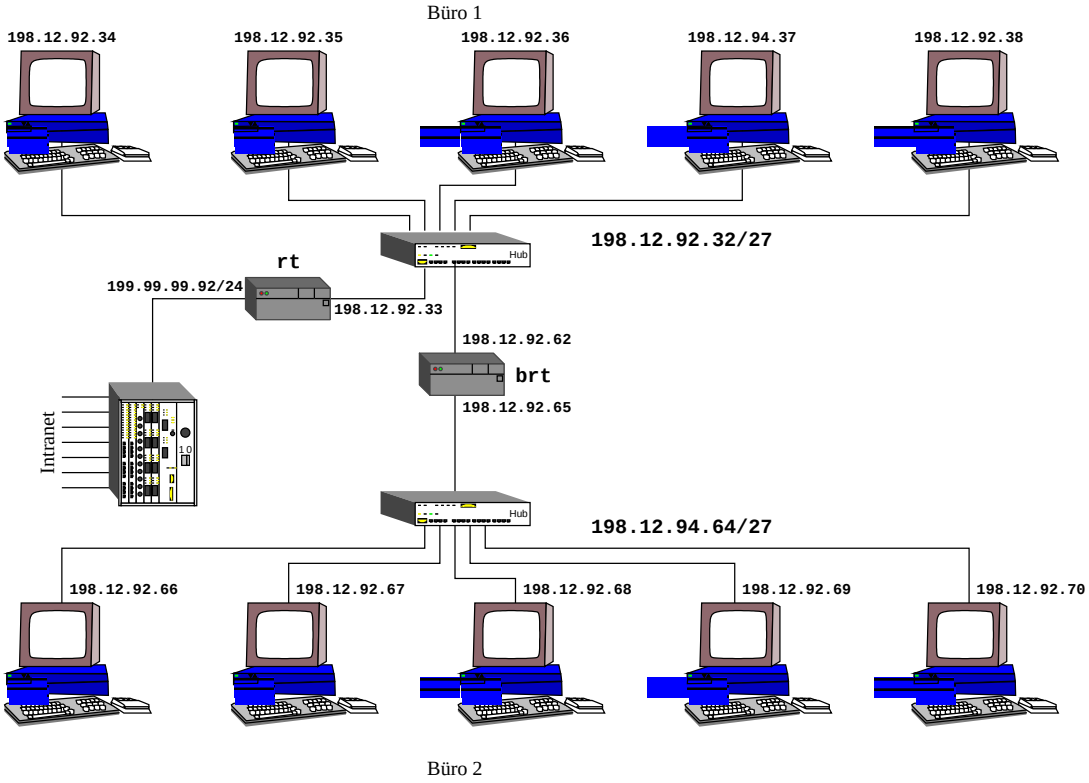

Abbildung 24.2: Netzwerkplan von zwei Büros mit Subnetting

24.2.4 Analyse

Zuerst die Negativen :

1. Werden es mehr als 6 Büros, wirds heikel. Eventuell kann das Problem dann durch eine weitere Verschiebung der Maske umgangen werden, sofern nicht mehr als $2^4 - 2 = 14$ IP-Addressen im Büro benötigt werden.
2. Das gleiche gilt umgekehrt

Und nun die Positiven :

1. Die Verschwendung von IP-Addressen im Unternehmen reduziert sich auf nur noch 256 Addressen, wobei jedoch noch 4 Büros ausgestattet werden könnten
2. Nur noch ein Routingeintrag für alle Router und Server

Das Ergebnis allerdings ist nicht so befriedigend ! Daher :

Kapitel 25

TOTAL SUBNETTING

25.1 Grundlagen

Man sollte grundsätzlich versuchen die Netzwerkaddressen im Baummodel zu verteilen. Damit reduzieren sich die Routingeinträge erheblich. Dazu verfügen die Router nur die Routen in die unteren Netzwerke und eine Default-Route zum oberen Router. Durch die Route-Redirect Meldungen sammelt sich der Router die Routen in die Netze gleicher Ebene selbst.

Dazu ist es vielleicht hilfreich sich die Grafik 25.1 und die nachfolgenden Routingtabellen der Router anzuschaun.

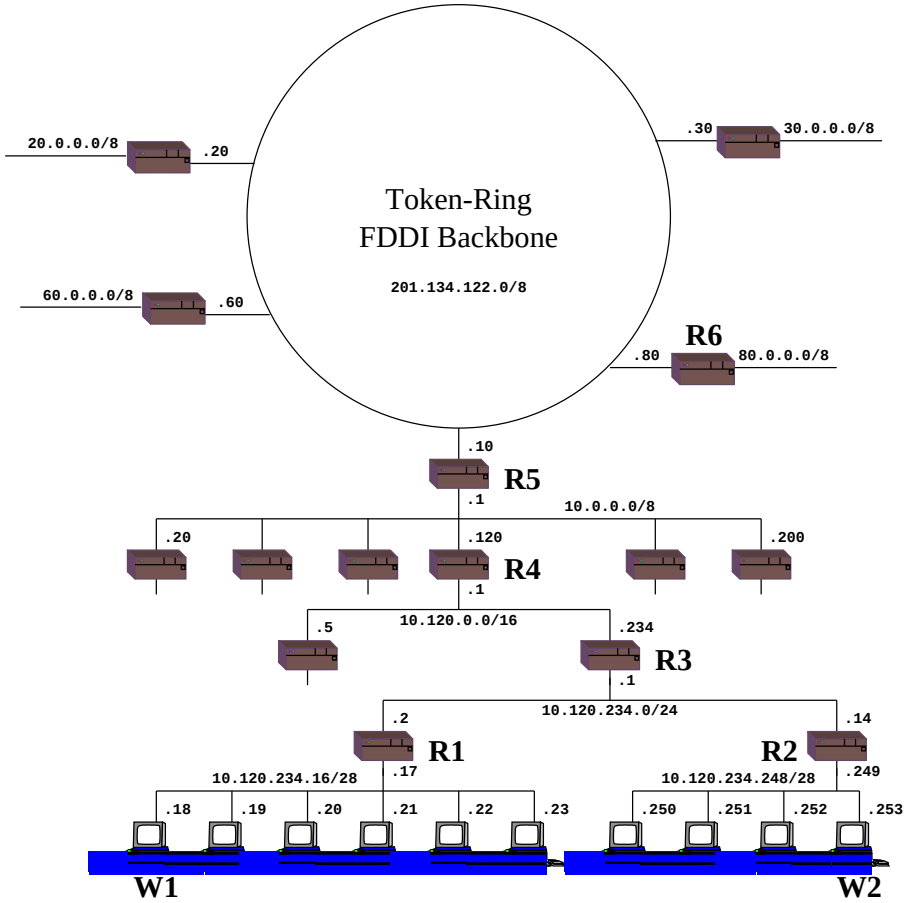

Abbildung 25.1: Ein Baumartiges Netzwerk

Wir müssen alle Register ziehen !

25.2 Workstation W1

Die Arbeitsstation W1 befindet sich im Subnet `10.120.234.16/28` und hat die IP Addresse `10.120.234.18`. Seine Routingtabelle beinhaltet nur zwei Einträge. Zum einen die Locale Route in sein eigenes Subnet und eine Defaultroute zum Router R1. Siehe Tabelle 25.1

Tabelle 25.1: Routingtabelle für Workstation W1

Destination	Gateway	Netmask	Metric
`10.120.234.16`	`10.120.234.18`	`255.255.255.240`	0
`0.0.0.0`	`10.120.234.17`	`0.0.0.0`	1

25.3 Workstation W2

Die Arbeitsstation W2 befindet sich im benachbartem Subnet `10.120.234.248/28` und hat die IP Addresse `10.120.234.252`. Seine Routingtabelle beinhaltet auch nur zwei Einträge. Zum einen die Locale Route in sein eigenes Subnet und eine Defaultroute zum R2. Siehe Tabelle 25.2

Tabelle 25.2: Routingtabelle für Workstation W1

Destination	Gateway	Netmask	Metric
`10.120.234.248`	`10.120.234.252`	`255.255.255.240`	0
`0.0.0.0`	`10.120.234.249`	`0.0.0.0`	1

25.4 Router R1

Jetzt wird es interessant, wir kosten nun die Fähigkeit des Rechnens voll aus.

Der R1 ist in zwei Netzwerken zu Hause. Einmal in dem Netzwerk wo sich W1 befindet und einmal das Netzwerk wo sich auch R2 und R3 befindet. Mit 100%-iger Sicherheit können wir sagen das er eine Locale-Route in das Subnetz `10.120.234.16/28` benötigt, also zum W1 hin. Wie sieht jedoch die Route zum R3 hin aus ? Auf dem Plan steht `10.120.234.0/24`. Wenn jedoch diese Daten genommen werden kommt es zu einem Mega-Kill-Effekt auf R1. Mal angenommen man erstellt eine Route in das `10.120.234.0/24` Netzwerk mit dem Gateway `10.120.234.2`. Wenn man nun aber zum W1 will, bekommt man spätestens am R1 eine Fehlermeldung ! Wieso ? Nun R1 nimmt die *Zieladdresse* und AND'et die mit der Maske `/24`. Das Ergebinis : $10.120.234.18 AND 255.255.255.0 = 10.120.234.0$. Da nun das Ergebnis mit dem *Destination Network* der Routingtabelle übereinpasst sendet er das Paket an sein Gateway `10.120.234.2`. Aufmerksame Leser haben bemerkt : Falsche Richtung ! Der W1 wird am falschen Kabel gesucht.

Wer glaubt das war's .... der täuscht sich leider gewaltig :

Es geht weiter. Da R1 bemerkt hat das z.B. R3 ihm ein Paket gesendet hat, welches er hätte selbst ausliefern können, sendet R1 dem R3 ein Route-Redirect. Danach sendet R3 die Pakete an W1 automatisch ins falsche Kabel. Nur zu blöd das der Netzwerkadministrator dann versucht den Fehler am R1 zu korrigieren. Er muß zusätzlich R3 neu booten !

D.h. die Netzwerkaddresse, wo sich R1, R2 und R3 sich befindet, ist nicht wie im Plan die `10.120.234.0/24` sondern eine andere. Es ist das NULL'te Subnetz vom W1 bzw. W2, also

10.120.234.0/28. Warum aber schreibt man dann so ein Müll in den Netzwerkplan ? Die gedruckte Netzwerkaddresse ist für außenstehende Netzwerkadministratoren interessant, damit die sehen wo sich das Netzwerk befindet. Please remember : Was hinter einem Router passiert ist dem, der davor steht, vollkommen egal ! Wir brauchen also noch eine Route in das andere Localenetzwerk.

Wie kommt der R1 jetzt in das Subnetz wo sich W2 befindet ? Es gibt hier zwei Möglichkeiten. Zum einen kann man dem Router nun eine Route in das Subnetz verpassen und gut ist. Nachteil ist nur das die Wartung des Routers auf, in diesem Fall maximal 13 Routingeinträge, anschwillt. Zum zweiten stellt sich die Frage, wer muß die Routen in die benachbarten Subnetze unbedingt kennen ? Klar R3 der muß alle kennen. Leider kann man sich eine Routingtabelle nicht Downloaden. Was nun ? Total simpel eigendlich :

Wir senden alle restlichen Pakete an R3, er weiss wie es weitergeht !

Bekommt nun der R3 ein Paket von R1 welches er an R2 ausliefern soll, dann bemerkt R3 automatisch das R1 das Paket gleich an R2 hätte senden können. Und genau da kommt das Route-Redirect von TCP/IP zum Einsatz. R3 sendet R1 nun eine Nachricht : *Lass das ! Sende es gleich zum R2 !*. R1 gibt sich geschlagen und handelt entsprechend. Der Eintrag wird automatisch mit in die Routingtabelle aufgenommen. Problem gelöst ! Also nur lächerliche 3 Routingeinträge, siehe Tabelle 25.3

Tabelle 25.3: Routingtabelle für Router R1

Destination	Gateway	Netmask	Metric
10.120.234.0	10.120.234.2	255.255.255.240	0
10.120.234.16	10.120.234.17	255.255.255.240	0
0.0.0.0	10.120.234.1	0.0.0.0	1

25.5 Router R2

Bei dem R2 verhält sich alles wie bei R1, die Routingtabelle 25.4 zeigt nur leichte Unterschiede.

Tabelle 25.4: Routingtabelle für Router R2

Destination	Gateway	Netmask	Metric
10.120.234.0	10.120.234.14	255.255.255.240	0
10.120.234.248	10.120.234.249	255.255.255.240	0
0.0.0.0	10.120.234.1	0.0.0.0	1

25.6 Router R3

Der R3 ist der eigendliche Subnetzrouter für die Subnetze. Er benötigt alle Routen in die unteren Netzwerke und in sein unteres Locales-Netz.

Aber hier ist VORSICHT angesagt. Die Route in das angebliche 10.120.0.0/16 darf nicht so verweinbart werden sonst kommt es wieder zum Mega-Kill-Effekt (siehe oben). Auch hier ist das obere Netz das Subnetz NULL vom unteren.

Desweiteren gilt hier wieder die Ausnutzung des R4 als Routingmanager der benachbarten Netzwerke (siehe auch oben)

Somit kommen wir auf 2 Locale Routen, 1 Default Route und pro untenliegndes Subnet eine Route.

Bei dem Netzwerkplan kommen wir also auf ganze 5 Routingeinträge. Siehe Tabelle 25.5

Tabelle 25.5: Routingtabelle für Router R3

Destination	Gateway	Netmask	Metric
10.120.0.0	10.120.0.234	255.255.255.0	0
10.120.234.0	10.120.234.1	255.255.255.240	0
10.120.234.16	10.120.234.2	255.255.255.240	1
10.120.234.248	10.120.234.14	255.255.255.240	1
0.0.0.0	10.120.0.1	0.0.0.0	1

25.7 Router R4

Auch hier gilt alles was beim R3 gilt. Bei diesen Routingtabellen 25.6 allerdingt sieht man jetzt erst die Bedeutung der im Netzwerkplan abgedruckten Netzwerkaddressen. Hier kommen wir auf lächerliche 4 Einträge.

Tabelle 25.6: Routingtabelle für Router R4

Destination	Gateway	Netmask	Metric
10.0.0.0	10.0.0.120	255.255.255.0	0
10.120.0.0	10.120.0.1	255.255.255.0	0
10.120.234.0	10.120.0.234	255.255.255.0	1
0.0.0.0	10.0.0.1	0.0.0.0	1

25.8 Router R5

Auch hier gilt das was am R4 galt. Allerdingt sollte dieser Router im Token-Ring keine Defaultroute haben. Man kann zwar auch hier das Route-Redirect ausnutzen, aber im Angesicht der Topologie ist dieses Datenpaket ehr störend. Die Routingtabelle 25.7 hält sich auch hier noch in Grenzen.

Tabelle 25.7: Routingtabelle für Router R5

Destination	Gateway	Netmask	Metric
201.134.122.0	201.134.122.10	255.255.255.0	0
80.0.0.0	201.134.122.80	255.0.0.0	1
30.0.0.0	201.134.122.30	255.0.0.0	1
60.0.0.0	201.134.122.60	255.0.0.0	1
20.0.0.0	201.134.122.20	255.0.0.0	1
10.120.0.0	10.0.0.120	255.255.0.0	1

25.9 Router R6

siehe R5 nur anderst ebend.

25.10 Zusammenfassung

Zusammenfassend lässt sich sagen. Je mehr ein Netzwerk gut strukturiert ist desto weniger Routingeinträge und desto weniger Aufwand hat man. Man muß ebend nur das TCP/IP ausnutzen.

TEIL VII

DNS

- Die anfänge von Hostnamen
  - Aufbau eines Hostnamens
  - Die hosts Datei
- Praktikum
- Domainnamen
  - Organisation des DNS
  - Begriffe zu Domainnamen
- Das Nameserver Umfeld
  - Resolver
  - Resolution
  - Root Name Servers
- Zonen
  - Zonendateien
  - Resource Record Types
  - Zwei Arten von Zonendatein
- BIND Konfiguration
  - BIND 4 Konfiguration
  - BIND 8 Konfiguration
- BIND Implementation
  - Linux Allgemein
  - SuSE Linux

  - Solaris
- Resolver Konfiguration
  - Allgemein UNIX
  - Speziell SuSE-Linux
  - Speziell Solaris
- Praktikum
  - Subnetz Server
  - Masterserver

Kapitel 26

DIE ANFÄNGE VON HOSTNAMEN

Dieser Teil beschäftigt sich mit Hostnamen. Den Menschen fällt es meist schwer sich irgendwelche 32 bit Breite kombinationen zu merken, dafür wurde die IP Addressierung erfunden, doch 4 Zahlen zumerken erweist sich auch als schwierig, des wegen gibt es Hostnamen. Man muss sich dann nur noch um Namen kümmern, was im Endeffekt wesendlich schwieriger ist als eine 4 Stellige IP Addresse, naja .....

Als man TCP/IP entworfen hat kann auch dann irgendwann die Erkenntnis, dass es mit IP-Nummern schwer werden würde. Aus diesem Grunde entschied man sich den Hosts einen Namen zugeben. Das hat zugegeben auch ein paar Vorteile. Wenn man in einer Firma einen Printserver aufsetzt benötigt man nur noch den Namen des Servers und nicht mehr die IP-Addresse. Weiterhin wenn der Printserver ein anderer Host sein soll, dann braucht man nur die Zuordnung von IP-Addresse und Hostnamen ändern und keiner merkt den Unterschied. Der Hostname ist der gleiche, jedoch ist die IP-Addresse dahinter eine andere.

Am Anfang gab es eine einzige HOSTS.TXT die auf einem ganz bestimmten Server lag, in der alle damaligen Host verzeichnet waren. Wer auf den aktuellen Stand sein wollte musste sich die HOSTS.TXT via FTP[1] abholen. Das ging solange gut bis es dann die ersten grossen Änderungen gab. Das SRI[2]-NIC[3] gab sich geschlagen, die Ladezeiten wurden irgendwann zu hoch und die Administration war extrem Zeitraubend und es gab irgendwann mehrere HOSTS.TXT die teilweise eigens manipuliert waren. Dazu gibts zum Glück eine Abhilfe zu der wir noch kommen werden.

26.1 Aufbau eines Hostnamens

Ein Hostname kann aus den kleinen oder grossen Buchstaben sowie Ziffern bestehen. Zwischen kleinen und grossen Buchstaben wird nicht unterschieden.

```
<hostname> ::= <letter> <letdig>
  <letdig> ::= { <letter> | <digit> | <sonlet> } [ <letdig> ]
   <digit> ::= [0-9]
  <sonlet> ::= [-]
  <letter> ::= [A-Za-z]
```

oder als Regulärer Ausdruck :

```
  hostname ::= [A-Za-z][A-Za-z0-9-]*
```

Ein gültiger Hostnamen wäre TIG09, jedoch nicht 01SRV. Um jetzt einem Host einen Namen zu geben ist die hosts Datei relevant.

[1] FTP-File Transfer Protocol
[2] SRI-Standfort Research Institute
[3] NIC-Network Information Center

26.2 Die hosts Datei

Die hosts Datei ist eine locale Datenbank die einen Hostnamen in eine IP-Addresse, oder umgekert, wandelt. Der Standort im Dateisystem ist jedoch von Betriessystem zu Betriebssystem unterschiedlich :

Betriebssystem	Standort
UNIX	/etc/hosts
NOVELL	SYSTEM:ETC/HOSTS
OS/2 WARP SERVER 4.0	C:/MTP/ETC/HOSTS
WINDOWS NT	C:/WINNT/SYSTEM32/DRIVERS/ETC/HOSTS
WINDOWS 95	C:/WINDOWS/HOSTS

Tabelle 26.1: Standorte der hosts Datei

Da diese Datei auf jedem Host gepflegt werden muss gelten die Regeln wie beim alten HOSTS.TXT Prinzip. Jeder Host sollte die gleichen Namen für IP-Addressen verwenden. Aber kommen wir mal zur hosts :

1. Die hosts ist eine ASCII-Text Datei und kann mit jeden handelsüblichen `vi` bearbeitet werden
2. Jede Zeile der hosts $\equiv$ einen IP-Address $\rightarrow$ Hostnameneintrag
3. Kommentare werden durch ein Hash # am Zeilen Anfang eigeleitet und gelten bis zum Ende einer Zeile
4. Jede Zeile beinhaltet mindestens zwei Spalten. Als Spaltentrenner können TAB's oder Space's verwendet werden
5. Die Bedeutung der Spalten ist aus Tabelle 26.2 zu entnehmen

Tabelle 26.2: Spalteninhalte der hosts Datei

Feld	Beschreibung
1	IP-Addresse
2	Realer Hostname
3..	Optionale Alias Hostnamen

6. Grundsätzlich sollte der eigene Hostname mit der eigenen IP-Addresse aufgeführt werden
7. Die Datei wird für jeden Zugriff neu gelesen. Eine Änderung wirkt sich sofort aus

Wollen wir uns mal den TIGER anschauen und den seine hosts mal bearbeiten. Als

1. sollte immer das Loopback bekannt gegeben werden
2. sollte immer der eigene Hostname verzeichnet sein
3. sollten immer die Hosts eingetragen werden mit dem der Host selbst kontakt pflegt. Es können aber auch alle sein

Weiterhin definieren wir mal das der FALCON gleichzeitig auch unter dem Namen PRNSRV bekannt sein soll, PRNSRV wäre also ein Aliasname für FALCON . Als Ergebnis kann der Quelltext 26.2.1 herangezogen werden.

Quelltext 26.2.1 Beispiel hosts Datei für den TIGER Host

```
#
# hosts Datei fuer tiger
#
127.0.0.1     localhost
10.10.10.1    tiger

10.10.10.2    panther
10.10.10.3    cheetah

10.10.10.112 rt1-1
20.20.0.121  rt1-2

20.20.0.1     eagle
20.20.0.2     falcon      prnsrv

20.20.0.123  rt2-2
30.0.0.132   rt2-3

30.0.0.1      ant
30.0.0.2      bug
```

Jetzt ist es uns auf dem Host TIGER möglich den FALCON z.B. mittels einen `ping falcon` oder `ping prnsrv` anzu'ping'en. Das Betriebssystem liest einfach die hosts und sucht nach FALCON , wenn es den Eintrag gefunden hat gibt es dem Program `ping` die eigendliche IP-Addresse zurück. Auch `traceroute` verwendet nun die hosts um die IP-Addressen in Namen umzuwandeln.

Jedoch ist die Verwendung eines solchen Namens etwas eingeschränkt. Es dürfen keine doppelten Namen auftreten. Um dieses zu verhindern gibt es, dank Paul Mockapetris (1984), das DNS[4]

[4] DNS-Domain Name System

Kapitel 27

PRAKTIKUM

1. Sammeln Sie sich in jedem Subnetz zusammen und denken Sie sich Hostnamen aus. z.B. Subnetz 1 aus dem Bereich Tiere, das zweite Subnet aus den Bereich Städte etc. pp.
2. Nachdem Sie sich für Hostnamen entschieden haben konfiguriert jeder den Hostnamen an seinem Rechner. Benutzen Sie dazu unbedingt den YaST und booten Sie danach neu !
3. Tragen Sie danach in der /etc/hosts die restlichen Hostnamen aus Ihrem Subnetz ein.
4. Versuchen Sie nun die Hosts mittels des Hostnamens anzupingen.
5. Starten Sie ein `traceroute` zum Drucker. Die Router werden Ihnen mit der IP Addresse angezeigt.
6. Fügen Sie die IP Addressen der Router mit in die /etc/hosts ein. Sie können z.B. den Hostnamen RT23-0 verwenden für die eth0 am Router RT23
7. Starten Sie den `traceroute` erneut um die Eintragungen zu testen.
8. Tragen Sie die IP Addresse des Druckers ebenfalls in die /etc/hosts. Nennen Sie den Drucker dort einfach nur HP.
9. Ändern Sie nun die /etc/printcap dahingehend, daß Sie die IP Addresse durch den Hostnamen HP ersetzen.
10. Auch das sollten Sie prüfen, indem Sie z.B. die /etc/hosts ausdrucken. Funktioniert das nicht, dann Booten sie einfach nochmal .....

Kapitel 28

DOMAINNAMEN

Vor noch nicht allzulanger Zeit gab es Betriebssysteme die eine Datenspeicherung auf Datenträgern nur im flachen Modell zuliessen. Also keine Verzeichnisse oder Strukturen. Man musste also aufpassen und die Dateinamen entsprechend prüfen. Um das Problem zu vermeiden wurden Verzeichnisse zur Strukturierung erfunden. Das gleiche gilt für die Hostnamen. Die Grafik 28.1 stellt ein UNIX Dateiensystem einer DNS Datenbank gegenüber. [RFC 1034][1]

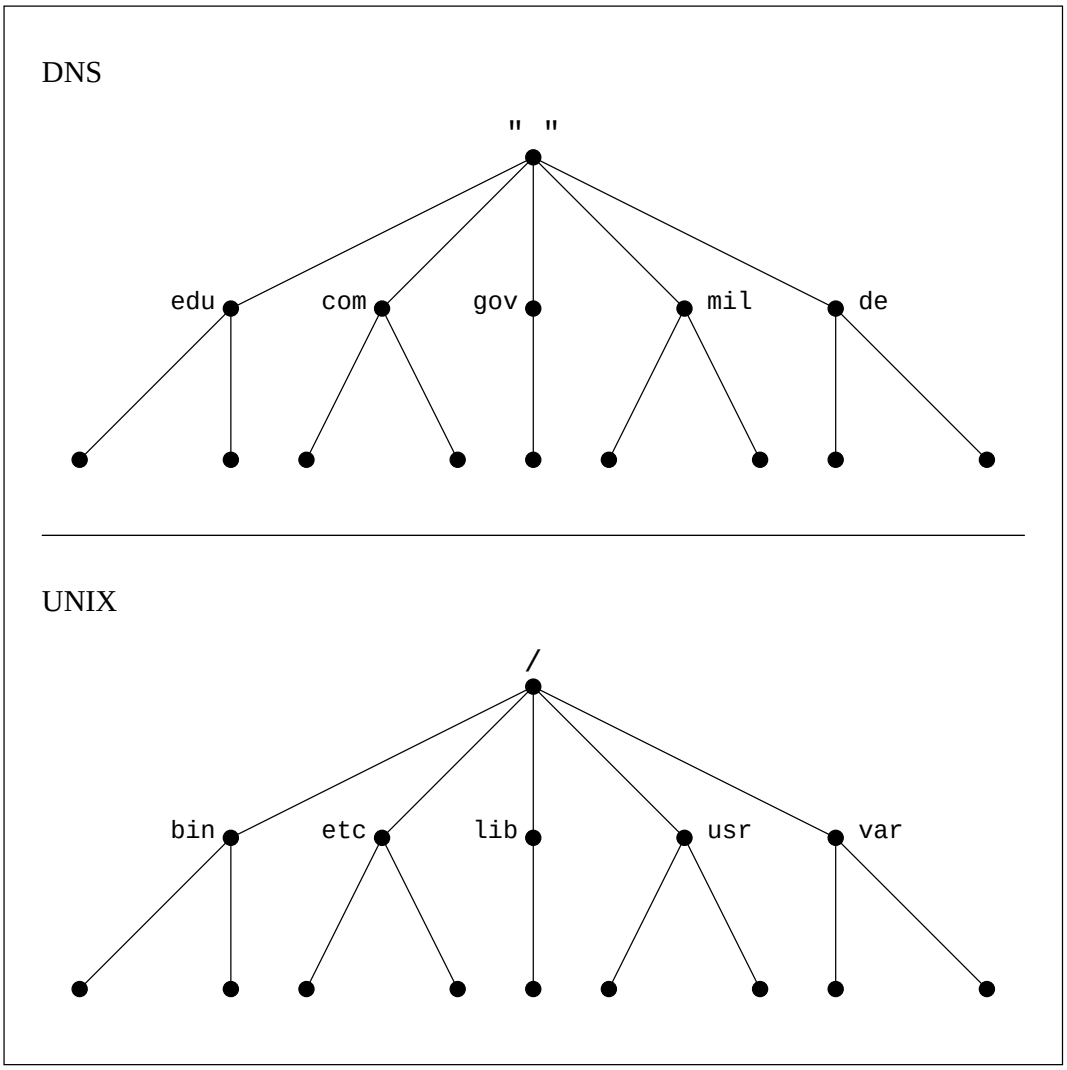

Abbildung 28.1: Gegenüberstellung DNS und UNIX System

Unter einem UNIX Dateisystem werden die einzelnen Stufen durch ein ”/”(Slash) getrennt. Unter

[1] RFC-1034 Domain Names - Concepts and Facilities

DNS werden die einzelnen Stufen durch ein ”.Punkt getrennt. Aber unter DNS geht man in der Beschreibung von unten nach oben und unter UNIX von oben nach unten, um einen Namen zu definieren. Siehe Grafik 28.2.

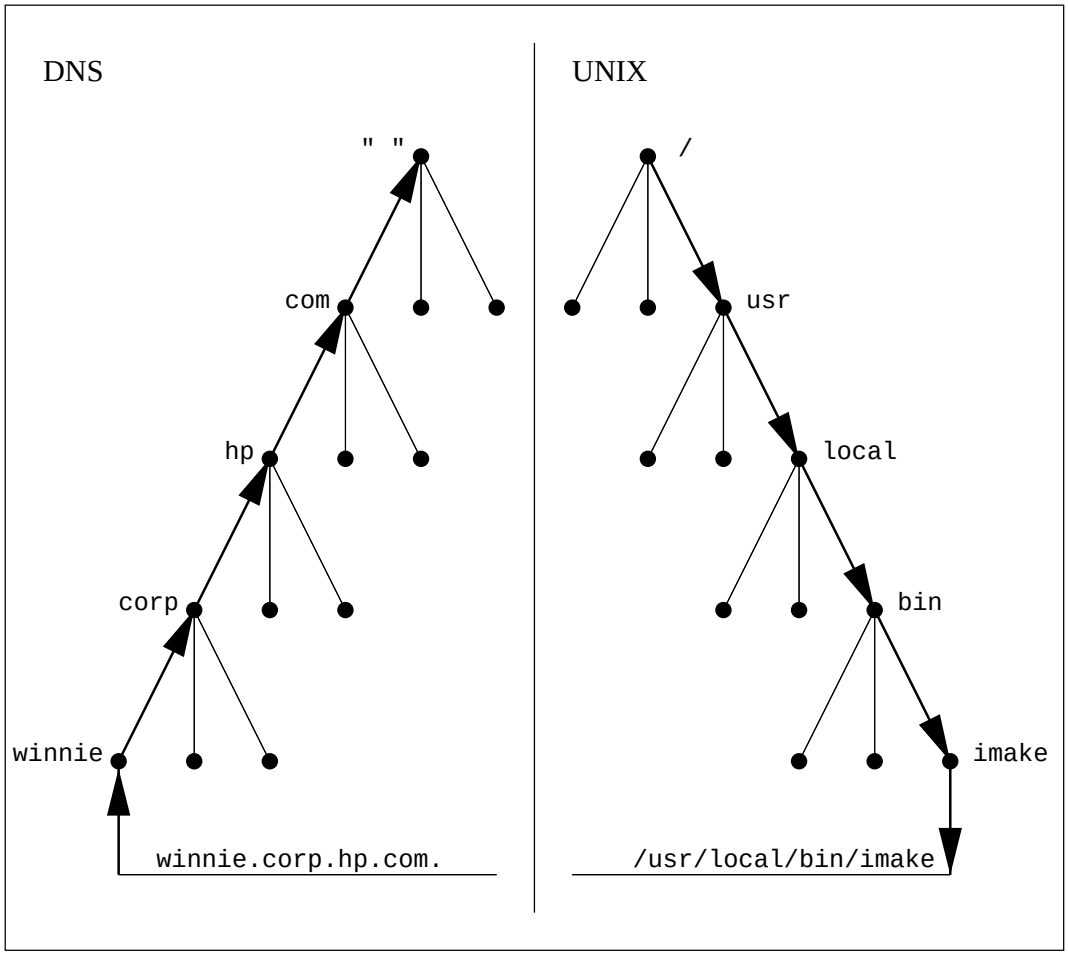

Abbildung 28.2: Gegenüberstellung der Namenslesung zwischen UNIX und DNS

Man beachte den abschliessenden Punkt hinter dem DNS Namen, er gibt den Absoluten Path zum Host an. Fast alle Programme kommen jedoch ohne diesen Punkt aus, weil im DNS es keine möglichkeit gibt eine ebene hochzugehen wie unter UNIX mittels .., sprich relative Angaben.

Und mit diesem Prinzip ist es nun möglich einen Namen zweimal zu vergeben, jedoch nicht innerhalb einer Domain, wie die Grafik 28.3 auf der nächsten Seite zeigt.

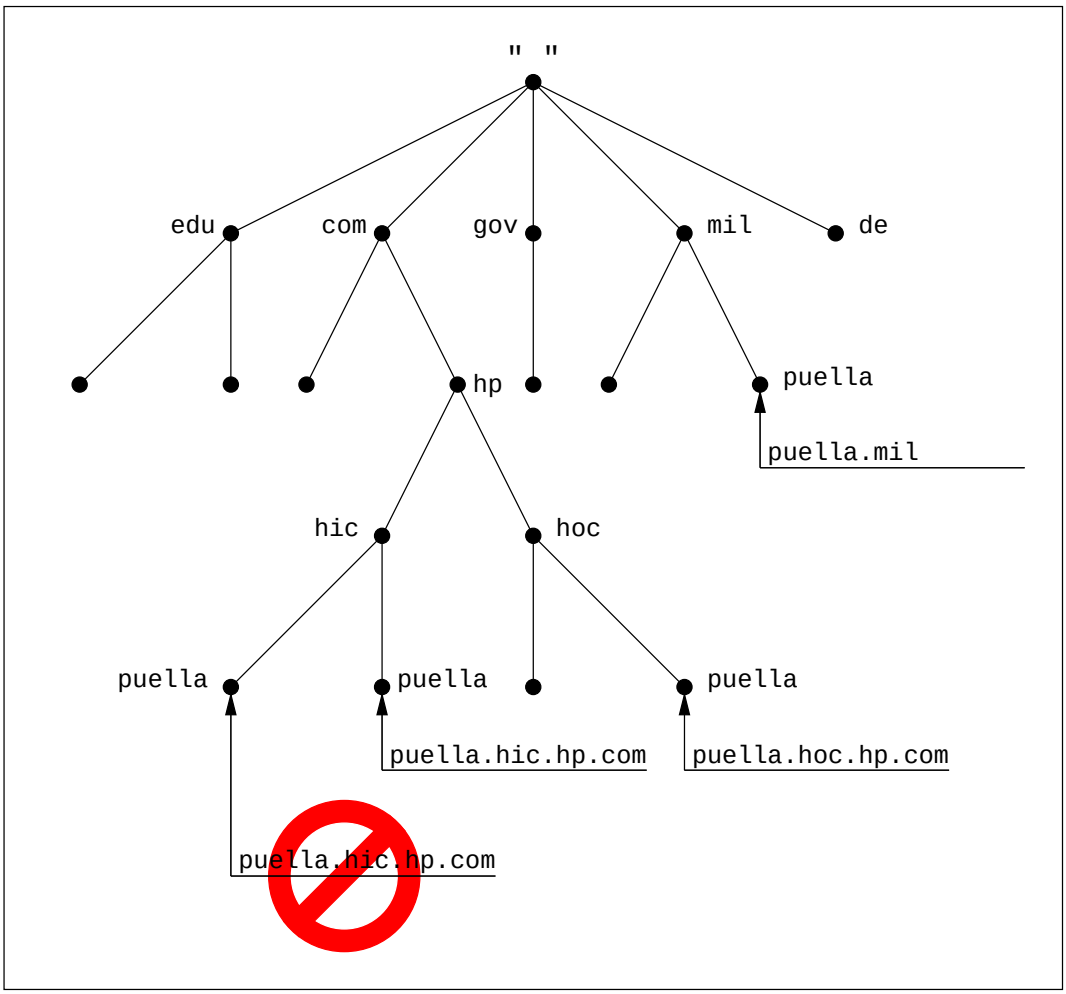

Abbildung 28.3: Doppelte Namen mit DNS

28.1 Organisation des DNS

Eine Domain ist eine logische Strukturierung. Eine Domain hat nichts mit der Anordnung der Hosts noch mit IP-Addressen zu tun. Am Beispiel unseres Netzwerkplanes z.B. könnte man nun hingehen und die Hosts TIGER, PANTHER und CHEETAH zu einem logischen Zusammenschluss, sprich einer Domain, zusammenzuführen. Dass die Host nun an einem Strang liegen ist Zufall, hat aber nichts mit der Domain zu tun. Wir überlegen uns nun einen passenden Domainnamen. Der Name CATS wäre sehr passend. Um diesen Namen nun benutzen zu können benötigen wir den Domainnamen der obenliegenden Struktur. Die ist z.B. CDI.DE. Somit können wir den FQDN[2] des Hosts TIGER bestimmen. Dieser wäre TIGER.CATS.CDI.DE. Somit bekommen wir einen DNS Baum wie die Grafik 28.4 auf der nächsten Seite uns zeigt.

[2] FQDN-Full Qualified Domain Name

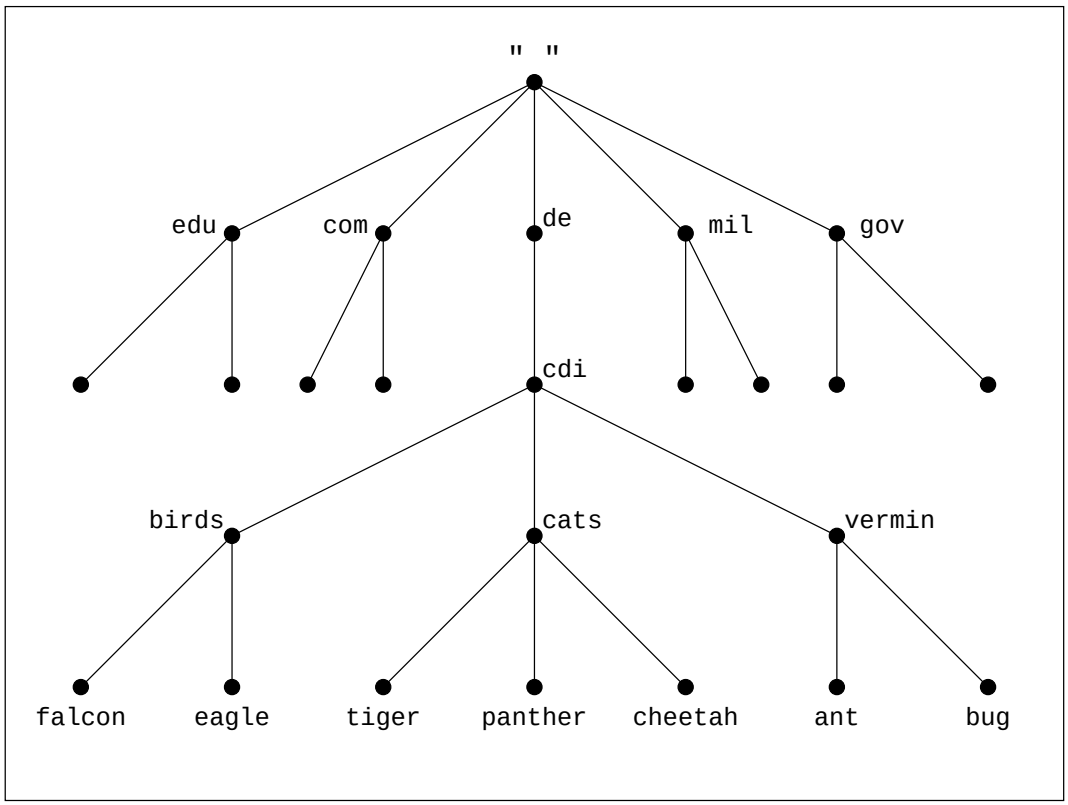

Abbildung 28.4: DNS Baum unseres Netzwerkes

Durch diesen Baum ergeben sich einige Domainen. Eine Domain wäre die DE Domain, eine weitere wäre die CDI.DE Domain. usw. siehe Grafik 28.5 auf der nächsten Seite.

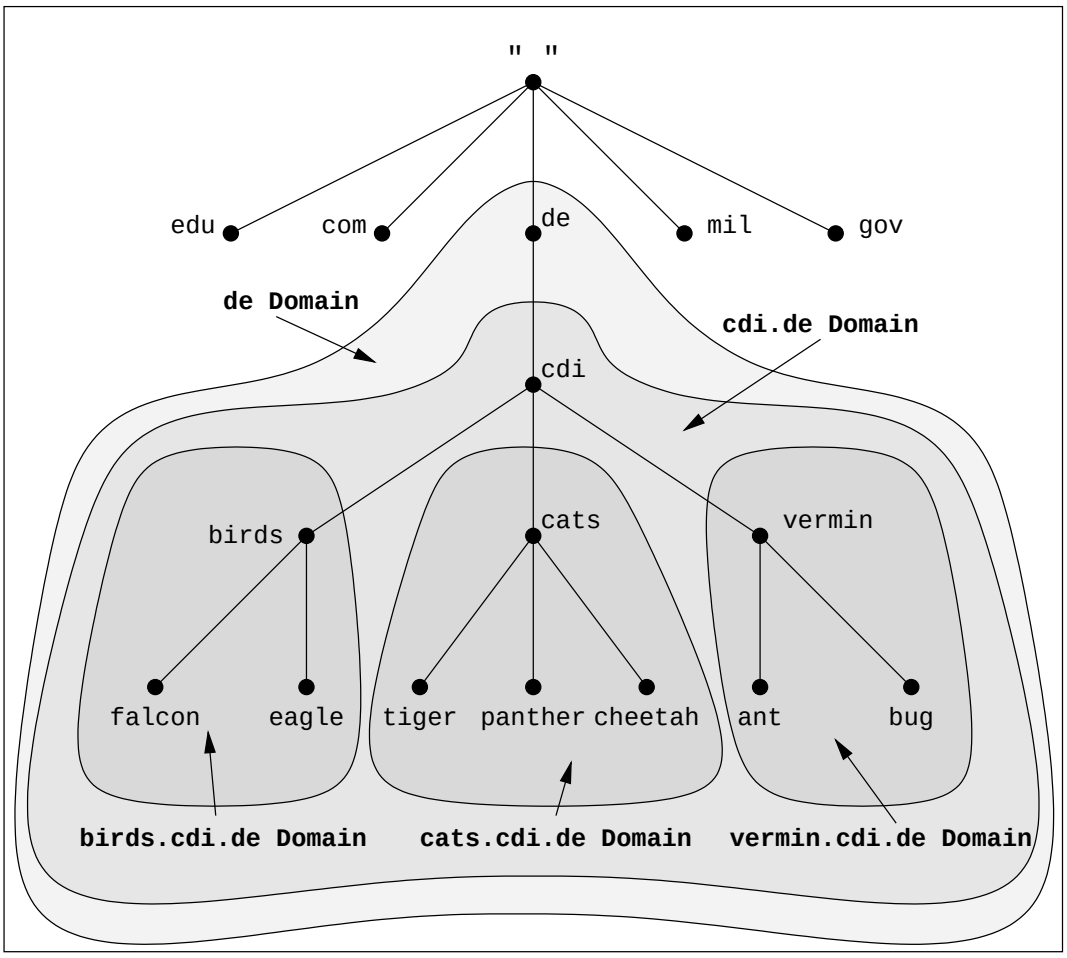

Abbildung 28.5: DNS Domainen unseres Netzwerkes

Jetzt könte man hingehen und die FQDN in der hosts Datei entsprechend eintragen. Jedoch ergeben sich dann wieder die bekannten Probleme der Synchonisierung und der Verwaltungsaufwand, etc. Man benötigt irgendwo eine Zentrale die alle Namen der Hosts speichert und die Namen bzw. die IP-Addressen auf verlangen rausrückt. Und soetwas gibt es auch, nennt sich Nameserver. Aber bevor wir zum Nameserver kommen noch einige Begriffserklärungen :

28.2 Begriffe zu Domainnamen

Es gibt einige Begriffe die wir im vorfeld behandeln müssen :

root Mit root wird die oberste Spitze des DNS-Baumes bezeichnet. Also der einfache Punkt

child Eine child Domain ist eine Domain eine Ebene tiefer

Top-Level-Domain Eine Top-Level-Domain ist ein child der root Domain. Es gibt einige standarisierte Top-Level-Domains.

First-Level-Domain Ein anderer Begriff für die Top-Level-Domain

Subdomain Eine Domain ist eine Subdomain, wenn es noch eine Domain oberhalb gibt. z.B. wäre CATS eine Subdomain von CDI.DE. Aber auch CDI ist eine Subdomain von DE

Hostname Der eigendliche Hostname, wie TIGER z.B

FQDN Der volle Name des Host's wie TIGER.CATS.CDI.DE

Kapitel 29

DAS NAMESERVER UMFELD

BIND[1] ist wohl der bekannteste Nameserver, sowohl für UNIX als auch für OS/2. Unter WINDOWS NT wird etwas ähnliches verwendet. BIND ist ein Programm das als Server fungiert. BIND bearbeitet anfragen aus dem Netz. BIND hat die Aufgabe das DNS abzubilden. Man spricht im Allgemeinen vom DNS-Server oder Nameservern. [RFC 1035][2]

29.1 Resolver

Ein Resolver ist ein Client. Der Resolver fragt den Nameserver entweder nach einen Namen oder nach einer IP-Addresse. Der Resolver wartet dann auf eine Antwort des Nameservers. Der Mechanismus des Frage-Antwort Spiels ist bereits fertig in der Standardbibliothek abgelegt. Und jedes Programm das einen Hostnamen zu einer IP-Addresse wandeln will, benutzt diese eine Funktion der `stdlib`. Diese Funktion erfüllt gleichzeitig mehrere Anforderungen, zum einen prüft die Funktion die `hosts` und wenn die Funktion dort nicht fündig geworden ist, fragt sie über das Netzwerk einen Nameserver. Das Programm selbst bekommt das nicht mit, es bekommt von der Funktion nur eine entsprechende Antwort. Zur Konfiguration eines Resolvers werden wir noch kommen.

29.2 Resolution

Eine Resolution ist eigendlich genau das welches das Wort bereits schon aussagt. Eine Resolution steht für die erfolgreiche Auflösung eines Namen oder einer IP-Addresse in eine IP-Addresse bzw. in einen Namen.

29.3 Root Name Servers

Diese Nameserver stehen im Baum ganz oben. Nur diese Nameserver kennen die Nameserver der Top-Level-Domains. Im Jahr 1992 wurden ca. 20.000 Anfragen pro Stunde gezählt (6 pro sekunde). Der Root-Server des InterNIC NS.INTERNIC.NET zählte ganze 255.600 Anfragen pro Stunde (71 pro sekunde). Und das war 1992, heute sieht das wohl noch etwas krasser aus. Warum es soviele Anfragen gibt ist einfach. Mal angenommen Sie sitzen vor TIGER und wollen eine Verbindung mit GIRIGIRI.SUPPORT.HP.COM aufnehmen. Sie benötigen dazu die IP-Addresse. Die folgende Aufzählung wird von der Grafik 29.1 auf der nächsten Seite erläutert :

[1] `BIND`-Berkeley Internet Name Domain

[2] `RFC-1035` Domain Names - Implementation and Specification

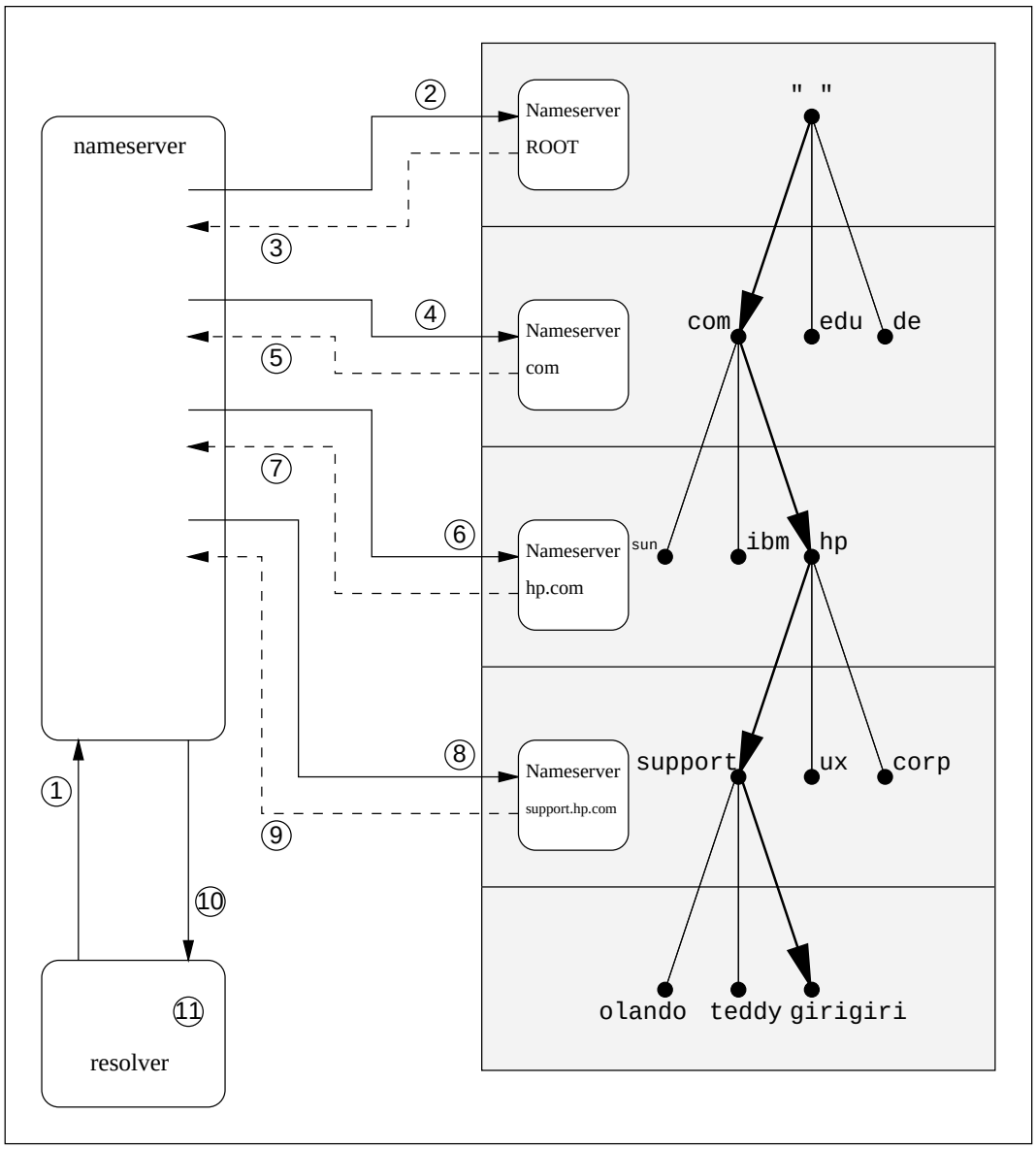

Abbildung 29.1: Resolution einer Anfrage

1. TIGER fragt den localen Nameserver, z.B. FALCON
2. FALCON hat keine Ahnung und fragt ein Root-Nameserver
3. Der Root-Nameserver gibt FALCON die Addresse des Nameservers für die Domain COM zurück
4. FALCON verwendet nun die Addresse und fragt den COM-Nameserver
5. Der hat auch keine Ahnung und gibt die Addresse des Nameservers der Domain HP zurück
6. FALCON fragt nun den Nameserver der Domain HP.COM
7. Wie sollte es auch anderst sein. Der hat auch kein Plan und sendet die Addresse des Nameservers der Domain SUPPORT.HP.COM zurück

8. FALCON (nerv) fragt nun den Nameserver der Domain SUPPORT.HP.COM
9. Der Nameserver kennt den Host GIRIGIRI und sendet FALCON die IP-Addresse zuück
10. FALCON ist gluecklich und gibt TIGER die IP-Addresse von GIRIGIRI.SUPPORT.HP.COM zurück
11. TIGER baut Verbindung auf

Kapitel 30

ZONEN

Eine Zone ist eine Verwaltungseinheit eines Nameservers. Eine Zone kann eine oder mehrere Domains beinhalten. Beinhaltet eine Zone eine Domain, dann muß die Zone die komplette Domain managen. Eine Trennung von einer Domain auf mehrere Zonen ist nicht möglich.

Für jede Zone muß ein Nameserver definiert sein. Ein Nameserver kann mehrer Zonen führen. Das Trennen von Zonen auf mehrere Nameserver ist nicht möglich. Jedoch kann eine oder mehrere Zonen von verschiedenen Nameservern geführt werden, Backupstrategie.

Um eine Zone zu definieren benötigt man sogenannte Zonendateien :

30.1 Zonendateien

30.1.1 Aufbau eines Resource Records

Eine Zonendatei ist eine gewöhnliche ASCII Textdatei. Diese Zonendatei kann ein oder mehrere Domains beinhalten. Um die Zonendatei entsprechend zu konfigurieren, bedient sich der Nameserver sogenannten RR[1]'s. Ein RR hat immer den folgenden Aufbau :

```
name    [ttl]    class    record-type    record-specific-data
```

Tabelle 30.1: Felder im RR

Feld	Beschreibung
name	Steht für ein Name in einer Domain. Wird kein Name angegeben wird der Name aus dem vorherigen RR genommen.
ttl	Angabe der Zeit in Sekunden wie lange der Eintrag im Cache des Nameservers verbleiben soll. Wird die Angabe weggelassen, gilt die *minimum* Angabe im `SOA` RR
class	Angabe der Protocolclass. Zur Zeit wird nur `IN` für TCP/IP unterstützt
record-type	In diesem Feld wird der Typ des RR's angegeben. Es gibt verschiedene Typen die weiter unten behandelt werden
record-specific-data	Daten die zu diesem Namen für den entsprechenden Typ passen. Auch hier siehe unten

Die RR's haben auch ganz bestimmte Sonderzeichen.

30.1.2 Sonderzeichen im RR

[1] RR-Resource Record

Tabelle 30.2: Sonderzeichen in Zonendateien

Feld	Beschreibung
.	Ein freistehender Punkt im Namensfeld steht für die aktuelle Domain
@	Ein freistehender Klammeraffe im Namensfeld steht für den aktuellen Zonennamen (Origin)
..	Zwei freistehene Punkte im Namensfeld stehen für den Namen der ROOT Domain
()	Klammern werden benutzt um ein Zeilenterminator zu umgehen. Sprich eine Zeile in mehrere verteilen. Wird eigentlich nur im SOA RR gemacht
;	Ein Semikolon startet ein Kommentar bis zum Ende einer Zeile
*	Das Asterisk ist ein Wildcardzeichen und trifft auf alles zu

30.1.3 Controll Einträge in Zonendateien

Es gibt noch zwei mögliche Controllkommandos in Zonendateien die unbedingt in Spalte 1 beginnen müssen !

Man hat die Möglichkeit mittels $INCLUDE eine andere Datei an diesem Ort hinzuzufügen. Bei grossen Nameservern kann man so ständig wiederholte RR's in einer solchen Datei ablegen.

```
$INCLUDE filename
```

Weiterhin besteht die Möglichkeit ein Zonennamen innerhalb einer Zonendatei zu wechseln. Das macht sinn wenn mehrere Domains in einer einzigen Zonendatei verwaltet werden soll. Das gilt jedoch nicht für andere Zonen !

```
$ORIGIN domainname
```

Wurde kein ORIGIN angegeben gilt das Origin aus der Konfigurationsdatei des Nameserver.

30.2 Resource Record Types

Als Types eines RR"s kommen die folgenden Werte in Frage. Wobei weiter unter auf die Einzelne Bedeutung näher eingegengen wird.

Tabelle 30.3: Resource Record Typen

Feld	Beschreibung
SOA	Start of authority. Begin einer vertrauenswürdigen Zone
NS	Nameserver Eintrag
A	Umwandlung von einem Namen zu einer Internet Addresse
PTR	Umwandlung einer IP Addresse in einen Hostnamen
CNAME	Canonicalname (nickname) Aliasname
TXT	Textinformationen

WKS	Well-known services
HINFO	Host Informationen
MX	Mail Exchanger
AAAA	Hostnamen zu IPv6
RP	Resonsible Person, Ansprechperson -experimental-

30.2.1 SOA

Der SOA RR wird benötigt um eine Zone für den Nameserver zu regestrieren. Bei Zonentransfers von Nameserver zu Nameserver wird dieser RR benötigt um die folgenden RR's einzuordnen. SOA muß immer der erste RR sein. Er hat folgendes Format :

```
name  IN  SOA  ns-host  email (
        serial
        refresh
        retry
        expire
        ttl   )
```

Dazu muß man nun etwa folgendes zu sagen :

Tabelle 30.4: Parameter im SOA

Feld	Beschreibung
name	Der Name der Zone für diesen SOA. Der Zonenname **muß** mit einem Punkt abgeschlossen werden, da in den Zonendateien immer relativ zur ORIGIN gearbeitet wird
ns-host	FQDN des Nameservers der diese Zoneführt. Auch hier muß der FQDN mit einem Punkt abgeschlossen werden
email	Die EMail Addresse des verantwortlichen der Zone. Da in EMails der Klammeraffe vorkommt, der in Zonendateien eine besondere Bedeutung besitzt wird der Klammeraffe durch ein Punkt ersetzt.
serial	Seriennummer dieser Zonendatei. Andere Nameserver orientieren sich daran ob sie nun die Zone downloaden oder ob die eigene noch aktuell genug ist. Bei Änderungen in der Zonendatei sollte IMMER auch die Seriennummer geändert werden. Meist wird TagMonatJahr verwendet.
refresh	Diese Angabe in Sekunden gibt an wie lange ein Secundaryserver warten soll, bis er nochmal nachschaut um zu entscheiden ob ein Update der Zonendatei notwendig ist. Meist liegt der Wert so um die 7200, das sind ca. 2 Stunden.
retry	Diese Zeitangabe in Sekunden gibt an wie lange ein Secundaryserver nach einem Fehler warten soll bis er es nochmal versucht. Meist wird hier 3600 genommen, sprich 1 Stunde

expire	Diese Zeitangabe in Sekunden gibt an wie lange maximal ein Secundaryserver diese Daten benutzen soll. Nach ablauf dieser Zeit verfallten die Daten im Secundaryserver. Bei jedem Fehlerfreien Zugriff jedoch wird wieder angefangen zu zaehlen. Meist wird hier eine 432000 Eingetragen, was ca. 5 Tagen entspricht
ttl	Dieser Wert wird als TTL für die RR's gesetzt die keine TTL eingetragen haben. Meist wird 86400 für ein Tag eingetragen.

Ein solcher Eintrag kann ca. so aussehen :

Quelltext 30.2.1 Beispiel eines SOA RR in einer Zonendatei

```
@       IN      SOA     falcon.hurst.pnet. whurst.hurst.pnet. (
                        03071998        ; serial
                        28800           ; refresh
                        7200            ; retry
                        3600000         ; expire
                        86400 )         ; ttl
```

30.2.2 NS

Ein NS Eintrag ist ein Eintrag für ein Nameserver der eine Domain verwaltet. Der eigene Nameserver muß sich selbst in einer Zonendatei wieder finden, ansonsten bekommt man nur Fehlermeldungen. Ein NS-Eintrag ist wie folgt aufgebaut :

```
name [ttl] IN NS ns-name
```

Tabelle 30.5: Parameter im NS

Feld	Beschreibung
name	Name der Domain die ein Nameverver führt
ns-name	FQDN des Nameservers der diese Domain führt

Quelltext 30.2.2 Beispiel eines NS RR in einer Zonendatei

```
@         IN NS falcon.hurst.pnet.    ; Das bin ich selbst
xterm     IN NS ns.xterm.hurst.pnet.  ; Domain xterm fuehrt der Server ns.xterm
win       IN NS ant.win.hurst.pnet.   ; Domain win.hurst.pnet f"uhrt ant
```

30.2.3 A

Ein A Eintrag wird benötigt um einen Hostnamen einer IP zuzuweisen. Hier jedoch ist eine IPv4 Addresse zu verwenden. IPv6 Addressen werden mit dem AAAA Eintrag getätigt.

```
name IN A ipv4
```

Tabelle 30.6: Parameter im A

Feld	Beschreibung
name	Der Name des Hosts. Dieser wird immer relativ zum aktuellen Origin angegeben. FQDN's sind nur beim BIND 4 erlaubt, der neue BIND 8 liefert eine Fehlermeldung, die jedoch umgangen werden kann.
ipv4	Die IP Addresse des Hosts

Quelltext 30.2.3 Beispiel eines A RR in einer Zonendatei

```
falcon    IN  A  10.65.13.1
ns.xterm  IN  A  10.65.11.1
tiger     IN  A  10.65.13.7
```

30.2.4 PTR

Ein PTR Eintrag wird benötigt um eine IP Addresse in einen Hostnamen zu wandeln. Unter UNIX ist dieses extrem wichtig, da Berechtigungen anhand der IP Addresse und des Hostnamens geklärt werden. PTR Einträte findet man jedoch nur in Revers-Lookup-Zonen (siehe weiter hinten)

```
ip  IN  PTR  name
```

Tabelle 30.7: Parameter im PTR

Feld	Beschreibung
ip	Angabe der IP
name	FQDN des Hosts

Quelltext 30.2.4 Beispiel eines PTR RR in einer Zonendatei

```
1.13    IN  PTR  falcon.hurst.pnet.
1.11    IN  PTR  ns.xterm.hurst.pnet.
7.13    IN  PTR  tiger.hurst.pnet.
```

30.2.5 CNAME

CNAME wird verwendet um ein Aliasnamen zu vereinbaren. Der Nameserver selbst sollte im SOA nie ein Alias sein.

```
name  IN  CNAME  alias
```

Tabelle 30.8: Parameter im CNAME

Feld	Beschreibung
name	Der original Name
alias	Der Aliasname

Quelltext 30.2.5 Beispiel eines CNAME RR in einer Zonendatei

```
puma   IN  A      10.65.13.202
puma   IN  CNAME  nis
       IN  CNAME  www.homes.hurst.pnet.
```

30.2.6 TXT

Mit TXT können Zusatzinformationen angegeben werden.

```
name  IN  TXT  infos
```

Tabelle 30.9: Parameter im TXT

Feld	Beschreibung
name	Name des Hosts
infos	Informationen in Doublequotes eingeschlossen

Quelltext 30.2.6 Beispiel eines TXT RR in einer Zonendatei

```
puma   IN  TXT  "Das ist mein NIS Server"
```

30.2.7 WKS

Mit WKS können in der DNS Datenbank Informationen über die Dienste der entsprechenden Hosts gespeichert werden.

```
name  IN  WKS  address  protocol  service-list
```

Tabelle 30.10: Parameter im WKS

Feld	Beschreibung
name	Der Name des Hosts
address	Die IP Addresse des Hosts
protocol	Kann TCP oder UDP sein
service-list	Liste durch Whitespaces getrennt der unterstützten Dienste

Quelltext 30.2.7 Beispiel eines WKS RR in einer Zonendatei

```
tiger  IN  WKS  10.65.13.7  TCP  telnet smtp ftp rexec rlogin finger
```

30.2.8 HINFO

Um nochmehr spielerei kümmert sich `HINFO` mit dem man die Hostspezifischen Daten angeben kann.

```
name  IN  HINFO  hardware  os
```

Tabelle 30.11: Parameter im HINFO

Feld	Beschreibung
name	Der Hostname
hardware	Angabe der Hardware
os	Angabe des Betriessystems

Quelltext 30.2.8 Beispiel eines HINFO RR in einer Zonendatei

```
puma  IN  HINFO  AMD/300  Linux
crow  IN  HINFO  "ULTRA SPARC 5" "Solaris 7"
```

30.2.9 MX

Der `MX` Eintrag ist für *smtp* wichtig. Mit einem MX Eintrag kann festgelegt werden an welchen Host die EMail eigentlich ausgeliefert werden soll. Es sollte immer ein MX Eintrag für die Domain existieren, da EMails eigentlich immer an Domains addressiert sind muß der MTA wissen an welchen Rechner er das Ding ausliefern soll.

```
name  IN  MX  prior  smtphost
```

Tabelle 30.12: Parameter im MX

Feld	Beschreibung
name	Hostname oder Domainnamen
prior	Numerische Prioritätenliste. Sollte ein MX ausfallen wird der nächste genommen.
smtphost	Der eigentliche SMTP Host

Quelltext 30.2.9 Beispiel eines MX RR in einer Zonendatei

```
hurst.pnet.   IN  MX  10  falcon.hurst.pnet.
              IN  MX  20  puma
www           IN  MX  10  tiger
xterm         IN  MX  10  mail.xterm
```

In diesem Beispiel wird eine EMail an die Domain `info@hurst.pnet` zu erst an FALCON gegeben, ist der FALCON krank, wird PUMA kontaktiert. EMails an den WWW.HURST.PNET werden dem TIGER anvertraut. Und alle EMails in die Subdomain XTERM.HURST.PNET werden an MAIL.XTERM.HURST.PNET weitergeleitet.

30.2.10 AAAA

Der AAAA Eintrag wird bei der verwendung von IPv6 Addressen benötigt. Siehe dazu [RFC 1886][2]

```
name  IN AAAA ipv6
```

Tabelle 30.13: Parameter im AAAA

Feld	Beschreibung
name	Hostname
ipv6	IPv6 Addresse [RFC 1884][3]

Quelltext 30.2.10 Beispiel eines AAAA RR in einer Zonendatei

```
crow  IN  AAAA  1080:0:0:0:8:800:200C:417A
```

30.3 Zwei Arten von Zonendatein

Man unterscheidet zwei Arten von Zonendateien.

Lookupzones werden Hostnamen zu IP Addressen gewandelt

[2] RFC-1886 DNS Extensions to support IP version 6
[3] RFC-1884 IPv6 Addressing Architecture

Revers-Lookupzones werden IP Addressen zu Hostnamen gewandelt

30.3.1 Lookupzones

Zu den normalen Lookupzonen brauchen wir nicht mehr so viel erklären. Hier ein Beispiel angepasst auf das Netzwerk. Wobei `falcon` der Master für BIRDS.CDI.DE. ist, TIGER für CATS.CDI.DE. und ANT für VERMIN.CDI.DE zuständig ist

Quelltext 30.3.1 Beispiel - Lookupzone von birds.cdi.de

```
birds.cdi.de.  IN  SOA  falcon.birds.cdi.de.  root.falcon.birds.cdi.de. (
                         031219999 28800 7200 3600000 86400 )
birds.cdi.de.  IN  NS   falcon.birds.cdi.de.
               IN  MX  10 falcon

falcon  IN  A     20.20.0.2
        IN  CNAME prnsrv
        IN  CNAME prn.cdi.de.

eagle   IN  A     20.20.0.1
```

Das nächste Beispiel ist nun verkürzt und definiert den TIGER als Nameserver und PANTHER für den SMTP Server der Domain

Quelltext 30.3.2 Beispiel - Lookupzone von cats.cdi.de

```
@  IN  SOA  tiger  root.tiger (
            031219999 28800 7200 3600000 86400 )
   IN  NS   tiger
   IN  MX  10 panther

tiger   IN A 10.10.10.1
panther IN A 10.10.10.2
cheetah IN A 10.10.10.3
```

In der Domain VERMIN.CDI.DE. stellt sich `ant` als Nameserver zur Verfügung.

Quelltext 30.3.3 Beispiel - Lookupzone von vermin.cdi.de

```
@  IN  SOA  ant administrator.ant (
            031219999 28800 7200 3600000 86400 )
   IN  NS   ant
   IN  MX 10  ant

ant IN A 30.0.0.1
bug IN A 30.0.0.2
```

Somit hat nun jede dieser Domains einen eigenen Nameserver.

Jetzt entsteht für die Clients jedoch ein extremer Nameserverkonfigurationsaufwand. Den man nun verhindern kann in dem man die Baumstruktur von DNS ausnutzt und ein Server definiert der die ganze Domain CDI.DE. führt und für die Subdomains die entsprechenden Nameserver dazu befragt. Genau das wird auch gemacht. Wo sich der CDI.DE. Master Nameserver befindet ist jedoch vollkommen unrelevant. Wir wählen man den CHEETAH.CATS.CDI.DE. aus :

Quelltext 30.3.4 Beispiel - Lookupzone von cdi.de auf cheetah

```
@   IN   SOA   cheetah.cats root.cheetah.cats (
               031219999 28800 7200 3600000 86400 )
    IN   NS    cheetah.cats
    IN   MX 10 eagle.birds.cdi.de. ; mails an cdi.de sollen an eagle

cheetah.cats IN A 10.10.10.3

; die domain birds erreicht man ueber den falcon
falcon.birds IN A  20.20.0.2
birds        IN NS falcon.birds

; die domain cats findet man am tiger
tiger.cats   IN A  10.10.10.1
cats         IN NS tiger.cats

; die domain vermin findet man am ant
ant.vermin   IN A  30.0.0.1
vermin       IN NS ant.vermin

; fuer das www definieren wir mal den bug
; die mails an www.cdi.de sollen jedoch zum ant
bug.vermin   IN A 30.0.0.2
             IN CNAME www
www          IN MX 10 ant.vermin
```

Jetzt können alle Clients den CHEETAH.CATS.CDI.DE als Nameserver eintragen. Dieser löst nun auch Addressen aus den anderen Domains auf, in dem er die Nameserver dort fragt.

30.3.2 Revers-Lookup-Zones

Eine RLZ[4] macht genau das wozu andere Zonen nicht in der Lage sind. Eine RLZ löst eine IP Addresse in ein Hostnamen um. Das ist wichtig. Beispiel : Ein Druckserver bekommt die Berechtigungen in der /etc/hosts.lpd. Dort werden Hostnamen eingetragen. Beim Verbindungsaufbau jedoch bekommt der Server nur die IP Addresse des Clients. Nun muß er die IP Addresse in ein Hostnamen umwandeln können um die Berechtigung zu prüfen.

Um auch ganz sicher zu gehen sendet der Server den Hostnamen noch einmal zur Auflösung und vergleicht die IP Addressen miteinander, den Vorgang nennt man dann Double-Revers-Lookup

[4] RLZ-Revers Lookup Zone

Um die IP Addressen jedoch sinnvoll zu Verwalten werden IP Addressen ebenfalls in einem Domain-Modell abgelegt. Das spart unnötige zweifach Konfiguration. Um IP Addressen jedoch in ein Baummodell, wie es DNS tut, muß die IP Addresse rückwährtz gelesen werden.

Zum Beispiel : Die IP 10.20.30.40 kann man auch wie folgt in einem Baumdiagramm wiedergeben : Die Addresse 40 liegt unterhalb des Nodes 30, der wiederum unterhalb des Nodes 20 liegt, der wiederum liegt unterhalb des Nodes 10.

In einer Verzeichnisstruktur könnte man es wie folgt deklarieren `mkdir -p /10/20/30/40`.

Um dieses Verfahren gefahrlos in den vorhandenen DNS Baum zu implementieren wurden die Domains IN-ADDR und ARPA erstellt. D.h. die oben benutzte Addresse kann man auch wiefolgt darstellen `40.30.20.10.in-addr.arpa.`

Dadurch ergibt sich z.B. nun die Möglichkeit eine Zone 30.20.10.IN-ADDR.ARPA. zu erstellen und in dieser Zone die Addressen 0-255 zu definieren. Der Zonenname ist dabei in einer Zonendatei das Origin.

Was nicht möglich ist, ist die Aufteilung wie es mit Subnetzen gemacht wird. Eine Zone kann sich nie über mehrere Nameserver hinweg bewegen. Man kann nicht sagen die Addressen 10.20.30.0 bis 10.20.30.127 macht Nameserver 1 und den Rest macht Nameserver 2. Man muss sich hier für einen Nameserver entscheiden.

Eine RLZ muß nicht unbedingt auch gleich der Subnetmask sein. Die Subnetmask ist hier vollkommen unrelevant.

Auf dem Nameserver in der BIRDS Domain mus weiterhin eine RLZ erstellt werden der die IP Addressen 20.20.0 anbietet :

Quelltext 30.3.5 Beispiel - RLZ von 0.20.20.in-addr.arpa

```
@ IN SOA  falcon.birds.cdi.de. root.falcon.birds.cdi.de. (
          031219999 28800 7200 3600000 86400 )
  IN NS falcon.birds.cdi.de.
1 IN PTR eagle.birds.cdi.de.
2 IN PTR falcon.birds.cdi.de.
```

Der Nameserver im 10’er Segment tut das gleiche :

Quelltext 30.3.6 Beispiel - RLZ von 10.10.10.in-addr.arpa

```
@ IN SOA  tiger.cats.cdi.de. root.tiger.cats.cdi.de. (
          031219999 28800 7200 3600000 86400 )
  IN NS tiger.cats.cdi.de.
1 IN PTR tiger.cats.cdi.de.
2 IN PTR panther.cats.cdi.de.
3 IN PTR cheetah.cats.cdi.de.
```

Der Nameserver im 30’er Segment tut das gleiche :

Quelltext 30.3.7 Beispiel - RLZ von 0.0.30.in-addr.arpa

```
@ IN SOA  ant.vermin.cdi.de. administrator.ant.vermin.cdi.de. (
          031219999 28800 7200 3600000 86400 )
  IN NS ant.vermin.cdi.de.
1 IN PTR ant.vermin.cdi.de.
2 IN PTR bug.vermin.cdi.de.
```

Um diese Zonen auf dem CHEETAH zu zentralisieren muß jedoch in diesem Beispiel anderst vorgegangen werden. Und zwar wird hier der Mechanismus des Sekundärserver benutzt. Das läd Zonendateien vom Master zu einem Sekundärserver hoch und stehen dort dann auch zur Verfügung.

Grund: Um, wie im oberen Beispiel, Domains zu zentralisieren muß die tiefste gemeinsame Domain gefunden werden. Im oberen Beispiel war es die CDI.DE. hier jedoch wäre es IN-ADDR.ARPA. Das Problem wäre nun das der Nameserver alle anderen IP-Addressen der Welt auch noch führen müsste ! Weil : Eine Zone kann nicht durch mehrere Nameserver geführt werden.

Kapitel 31

BIND KONFIGURATION

Zur Zeit gibt es zwei primär genutzte BIND Implementationen. Zum einen die Version 4 und zum anderen die Version 8. Beide werden anderst Konfiguriert. Die Zonendateien ändern sich dardurch nicht. Die V8 bietet erheblich-erweiterte Einstellungen zur Systemsicherheit. Die Konfiguration ist auf jedem System die gleiche.

31.1 BIND 4 Konfiguration

Als Konfigurationsdatei verwendet die Version 4 die Datei /etc/named.boot. Die Datei ist eine ASCII-Text datei und besteht aus Schluesselwörtern und dessen Bedeutungen.

31.1.1 directory

Angabe des Basisverzeichnisses für Zonendateien.

```
directory  basedir
```

Tabelle 31.1: Schlüsselwort `directory` in der /etc/named.boot

Feld	Beschreibung
basedir	Angabe des Verzeichnisses zur Aufbewahrung der Zonendateien. Im Normallfall ist hier /var/named angegeben. Empfehlung des Authors das basedirectory nach /etc/named zu setzen, somit kann die Konfiguration mit einem Rutsch gespeichert werden, in dem nur noch das /etc Verzeichniss benötigt wird. Wo es allerdings steht ist egal

Quelltext 31.1.1 Beispieleintrag des Schlüsselwortes `directory` in der /etc/named.boot

```
directory /etc/named
```

Somit wird das Basisverzeichniss auf /etc/named gesetzt.

31.1.2 primary

Angabe einer Primären Zone die dieser Nameserver verwalten soll.

```
primary  zone  filename
```

Tabelle 31.2: Schlüsselwort `primary` in der /etc/named.boot

Feld	Beschreibung
zone	Den Namen der Zone, ohne abschliesenden Punkt. Dieser Name wird als ORIGIN in der Zonendatei als default gesetzt, sofern keine andere ORIGIN Anweisung in der Zonendatei folgt.
filename	Der Dateiname der Zonendatei die zur *zone* passt. Der Dateiname wird immer relativ zum Basisverzeichniss angegeben

Quelltext 31.1.2 Beispieleintrag des Schlüsselwortes `primary` in der /etc/named.boot

```
primary hurst.pnet pz/db.hurst.pnet
```

Definierung der Zone HURST.PNET. Die Zonendatei ist die /etc/named/pz/db.hurst.pnet

31.1.3 secondary

Angabe einer Sekundärenzone. Die Primärzone hält ein anderer Nameserver. Der Nameserver führt ein Zonentransfer durch und legt die Zone als Datei ab.

```
secondary  zone  nsip  filename
```

Tabelle 31.3: Schlüsselwort `secondary` in der /etc/named.boot

Feld	Beschreibung
zone	Name der Zone die als Sekundärzone geführt werden soll. Dieser muß mit dem Zonennamen auf dem anderen Server übereinstimmen.
nsip	Die IP Addresse des Nameservers der diese Zone primär führt
filename	Name des Dateinamens wo die Daten der Zone abgelegt werden. Diese Datei wird bei jedem Zonentransfer neu erstellt. Änderungen an einer vorhandenen Datei wirken sich nicht aus. Der Dateiname ist relativ zum Basisverzeichniss

Quelltext 31.1.3 Beispieleintrag des Schlüsselwortes `secondary` in der /etc/named.boot

```
secondary t-online.de 195.199.22.4 sz/t-online.de
```

Läd die Zone T-ONLINE.DE. vom Nameserver mit der IP `192.199.22.4` runter und speichert dessen Zonendatei in /etc/named/sz/t-online.de

31.1.4 forwardes

Mit forwarders können Nameserver angegeben werden an die sich der Nameserver wenden soll wenn er selbst keine Antwort auf eine Anfrage liefern kann.

```
forwarders ns-list
```

Tabelle 31.4: Schlüsselwort `forwarders` in der /etc/named.boot

Feld	Beschreibung
ns-list	Liste, durch whitespaces getrennt, von Nameservern

Quelltext 31.1.4 Beispieleintrag des Schlüsselwortes `forwarders` in der /etc/named.boot

```
forwarders 189.53.63.1
```

Wenn der Nameserver keine Antwort hat wendet er sich an den Nameserver mit der IP `189.53.63.1`

31.1.5 slave

Wenn dieses Schlüsselwort angegeben ist leitet der Nameserver alles zu den Forwarders weiter und missachtet seine eigenen Zonen.

```
slave
```

Quelltext 31.1.5 Beispieleintrag des Schlüsselwortes `slave` in der /etc/named.boot

```
slave
```

Der Nameserver ist nun ein Slave

31.1.6 BIND 4 Beispiel Konfigurationen

Nun der FALCON.BIRDS.CDI.DE führt zwei Zonen die wollen Konfiguriert sein :

Quelltext 31.1.6 Beispielkonfiguration BIND 4 auf FALCON

```
directory /etc/named
primary birds.cdi.de pz/birds
primary 0.20.20.in-addr.arpa pz/0.20.20
```

Der TIGER.CATS.CDI.DE führt auch zwei Zonen :

Quelltext 31.1.7 Beispielkonfiguration BIND 4 auf TIGER

```
directory /etc/named
primary cats.cdi.de pz/cats
primary 10.10.10.in-addr.arpa pz/10.10.10
```

und der ANT.VERMIN.CDI.DE führt ebenfalls zwei Zonen :

Quelltext 31.1.8 Beispielkonfiguration BIND 4 auf ANT

```
directory /etc/named
primary vermin.cdi.de pz/vermin
primary 0.0.30.in-addr.arpa pz/0.0.30
```

Der einzige der jetzt noch fehlt ist der Master über alle anderen. Der CHEETAH.CATS.CDI.DE muß die RLZ der anderen als Secondary aufnehmen :

Quelltext 31.1.9 Beispielkonfiguration BIND 4 auf CHEETAH

```
directory /etc/named
primary cdi.de pz/cdi
secondary 10.10.10.in-addr.arpa 10.10.10.1 sz/10.10.10
secondary 0.20.20.in-addr.arpa 20.20.0.2 sz/0.20.20
secondary 0.0.30.in-addr.arpa 30.0.0.1 sz/0.0.30
```

Jetzt müssen nur doch die Nameserver gestartet werden und dann läuft alles.

31.2 BIND 8 Konfiguration

Ja

Kapitel 32

BIND IMPLEMENTATION

32.1 Linux Allgemein

Unter Linux kann entweder BIND4 oder BIND8 implementiert werden, jenach bedarf. Beie BIND's jedoch werden auf die gleiche Art und Weise Implementiert.

32.2 SuSE Linux

Unter der SuSE Linux existiert ein Modul `/sbin/init.d/named` welches zum Starten und Stoppen des Nameservers verwendet werden kann.

```
/sbin/init.d/named {start|stop}
```

Tabelle 32.1: Optionen zum named-Modul von SuSE

Option	Beschreibung
`start`	Dient zum Starten des Nameservers
`stop`	Stoppt den Nameserver

Anwendung :

```
falcon: dozent as root # /sbin/init.d/named stop
Shutting down name server
falcon: dozent as root # /sbin/init.d/named start
Starting name server.
falcon: dozent as root # tail /var/log/messages
Dec  5 18:06:03 falcon named[18871]: starting
Dec  5 18:06:03 falcon named[18871]: master zone "localnet" (IN) loaded  ...
Dec  5 18:06:03 falcon named[18871]: master zone "0.0.127.in-addr.arpa"  ...
Dec  5 18:06:03 falcon named[18871]: master zone "hurst.pnet" (IN) loade ...
Dec  5 18:06:03 falcon named[18871]: master zone "13.65.10.in-addr.arpa" ...
Dec  5 18:06:03 falcon named[18871]: listening on [127.0.0.1].53 (lo)
Dec  5 18:06:03 falcon named[18871]: listening on [10.65.13.1].53 (eth0)
Dec  5 18:06:03 falcon named[18871]: Forwarding source address is [0.0.0 ...
Dec  5 18:06:03 falcon named[18872]: Ready to answer queries.
falcon: dozent as root #
```

Bildschirmausschnitt 32.2.1: Stoppen und Starten des Nameservers unter SuSE

Wichtig ist die Kontrolle des Systemlog in /var/log/messages !

32.3 Solaris

Kapitel 33

RESOLVER KONFIGURATION

33.1 Allgemein UNIX

Unter Linux kann der Resolver über die Datei /etc/resolv.conf eingestellt werden. Diese Datei bietet einem die Möglichkeit bis zu 3 Nameserver einzutragen und auch eine Suchliste der Domains anzugeben.

Die Suchliste kann z.B. BIRDS.CDI.DE CATS.CDI.DE enthalten. Wird nun ein Hostname wie CHEETAH eingegeben wird zu erst CHEETAH.BIRDS.CDI.DE gesucht und dann CHEETAH.CATS.CDI.DE gesucht. Versucht man den ANT anzusprechen muß man ANT.VERMIN.CDI.DE eingeben.

Die Suchliste sollte so klein wie möglich ausfallen, das je mehr Sucheinträge desto mehr Nameserver anfragen. Meist wird nur die lokale Domain in der Suchliste aufgenommen.

Um die Einstellungen zu tätigen können die Schlüsselwörter `nameserver` und `searchlist` verwendet werden. z.B.:

Quelltext 33.1.1 Beispiel einer /etc/resolv.conf

```
search birds.cdi.de cdi.de
nameserver 10.10.10.1
nameserver 20.20.0.3
```

Die Reihenfolge des Zugriffs muß'unbedingt geprüft werden. Die dazugehörige Konfigurationsdatei ist die /etc/host.conf. Nicht bei nsswitch Betriebssystemen. Diese Datei sollte wie folgt aussehen :

Quelltext 33.1.2 Einträge in der /etc/host.conf um DNS Abfragen zu ermöglichen

```
order hosts bind
multi on
```

Besitzt das Betriebssystem den nsswitch Mechanismus muß die DNS Abfrage über die /etc/nsswitch.conf eingestellt werden. Aus diesen Betriebssystemen gibt es keine /etc/host.conf. Es ist sicher zu stellen daß in dieser Datei der folgende Eintrag existiert :

Quelltext 33.1.3 Einträge in der /etc/nsswitch.conf um DNS Abfragen zu ermöglichen

```
hosts: files dns
```

33.2 Speziell SuSE-Linux

Vorsicht: Die Einträge in den Dateien werden durch den YaST Konfigurationsmodus teilweise kaputt gemacht. Es sollten die entsprechenden Einstellungen in der /etc/rc.config getätigt werden. Das kann

z.B. auch heissen das man dort den Mechanismus des überschreibens ausgeschalten kann. Dazu dienen die folgenden Einstellungen :

Quelltext 33.2.1 Ausschalten des YaST Kaputmichmach in der /etc/rc.config

```
CREATE_RESOLVCONF=no
CHECK_ETC_HOSTS=no
BEAUTIFY_ETC_HOSTS=no
CREATE_HOSTCONF=no
```

Oder man überlässt alles den YaST. Entweder man machts im Menü oder selbst in der /etc/rc.config. Nach den Änderungen muß das Programm SuSEconfig gestartet werden.

Quelltext 33.2.2 YaST Konfiguration eines DNS Resolvers in der /etc/rc.config

```
CREATE_HOSTCONF="yes"
CREATE_RESOLVCONF=yes
SEARCHLIST="birds.cdi.de cdi.de"
NAMESERVER="10.10.10.1 20.20.0.3"
```

Konfigurieren Sie unbedingt beide Dateien. Also die /etc/host.conf und die /etc/nsswitch.conf

Man kann jedoch auch den YaST benutzen. Im Menu Administration des Systems versteckt sich der Punkt Nameserver Konfigurieren :

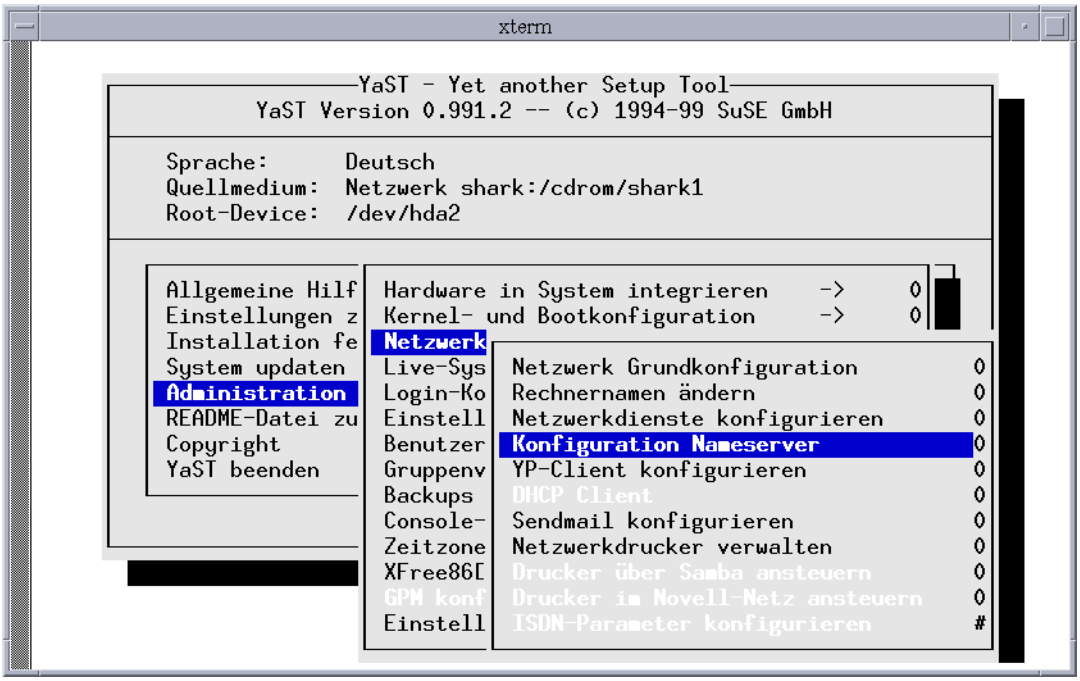

Abbildung 33.1: SuSE YaST Nameserverkonfigurationsmenuepunkt

Danach werden Sie gefragt ob Sie auf ein Nameserver zugreifen wollen, hmmm wenn wir nicht auf einen Nameserver zugreifen wollten, wieso hätten wir dann diesen Menüpunkt gewählt ?! Windose

lässt grüßen .... Danach bekommt man ein Dialog, Die Liste der Nameserver IP's entspricht dem Schlüssel `nameserver` in der /etc/resolv.conf. Die zweite Zeile Liste der Domains kommt mit dem Schlüssel `search` in der /etc/resolv.conf gleich :

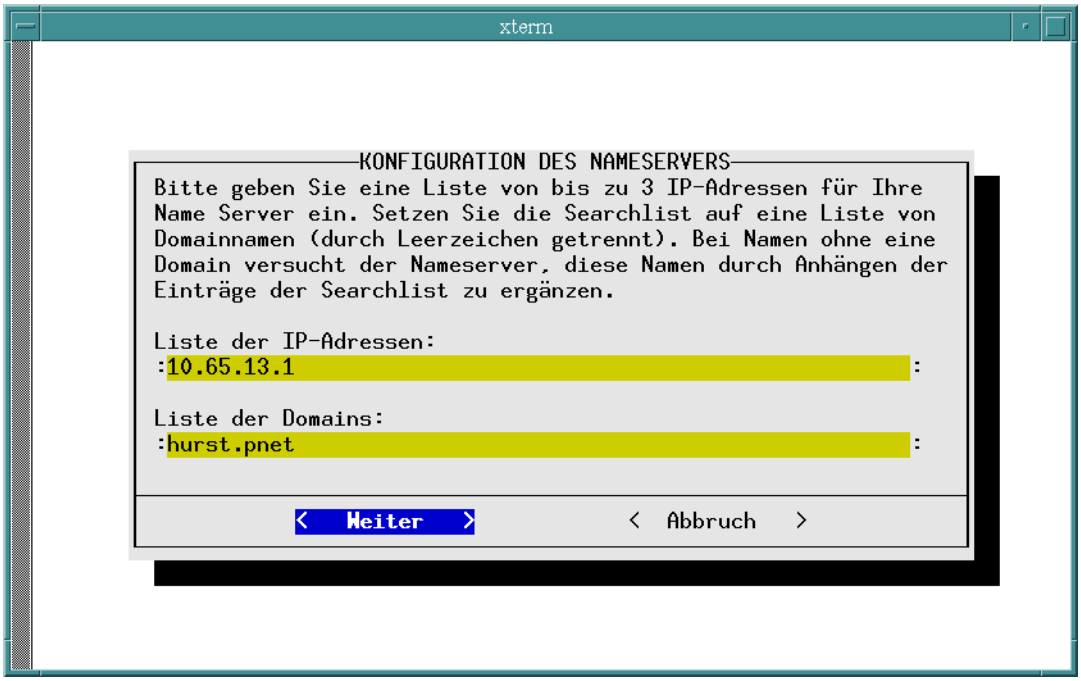

Abbildung 33.2: SuSE YaST Nameserkonfiguration

33.3 Speziell Solaris

Keine speziellen Spezialitäten zu beachten.

Kapitel 34

PRAKTIKUM

34.1 Subnetz Server

1. Eine Person aus jedem Subnetz findet sich beim grossen Treffen der Subnetze ein.
2. Bestimmen Sie für alle Subnetze ein Domainnamen der auf CDI.INTERN endet. Zum Beispiel : PLANETS.CDI.INTERN oder CITIES.CDI.INTERN oder DOGS.CDI.INTERN .....
3. Nach der Auflösung des Treffens der Subnetze, wird in jedem Subnetz ein Host bestimmt der Nameserver des Subnetzes spielen soll. Der Host muß für den weiteren Unterricht auch verfügbar sein.
4. Danach werden die Hostnamen aller IP Addressen innerhalb des Subnetzes festgelegt.
5. Erstellen Sie nun auf den Nameservern die Zonendatei für die Lookup-Zone, erstellen Sie auch die RLZ für Ihr Subnetz.
6. Konfigurieren Sie den Nameserver und testen Sie diesen. Achten Sie darauf daß der Hostname via YaST vorher korekt gesetzt wurde, und das alle überflüsdsigen Einträge aus der `/etc/hosts` verschwunden sind. Zum Testen kann entweder `nslookup` oder ein einfacher Ping dienen. Wichtig ist das der Nameserver selbst als Resolver Konfiguriert wird.
7. Nachdem der Test des Nameservers erfolgreich war, konfigurieren Sie nun die Clients entsprechend. Entfernen Sie alle nicht benötigten `/etc/hosts` Einträge und Konfigurieren Sie Hostnamen und Domain via YaST. Danach sollten Sie die Clients als Resolver konfigurieren.

34.2 Masterserver

1. Diese Übung dürfen Sie erst anfangen wenn die andere Übung von allen Subnetzen erfolgreich beendet wurde !
2. Definieren Sie nun ein Master-Nameserver der über allen Domains stehen soll. Dieser Host sollte dann genauso wenig wie die anderen mitten drin ausgeschaltet werden.
3. Auf diesem Nameserver soll nun die Lookupzone der Domain CDI.INTERN erstellt werden. Die Zone benötigt `NS` Einträge der anderen Domains.
4. Weiterhin muß der Nameserver das Netzwerk `10.40.11.xyz` als RLZ führen.
5. Starten Sie den Nameserver dort und testen diesen. Vergessen Sie nicht den Resolver entsprechend zu konfigurieren.
6. Nachdem die Funktionalität des Masters sichergestellt wurde, konfigurieren ALLE Clients den Resolver neu, in dem nun der Master benutzt wird.

7. Prüfen Sie die Konfiguration mittels ping.

TEIL VIII

INTERNET STANDARD DIENSTE

- Einführung
- Standardports
- inetd
  - Funktionsweise
  - Die Konfigurationsdatei file /etc/inetd.conflabel inetd-konfigdatei
  - Beispielkonfiguration
- Übersicht der einzelnen Dienste
  - echo
  - discard
  - systat
  - daytime
  - netstat
  - ftp
  - telnet
  - smtp
  - domain
  - bootp
  - tftp
  - gopher
  - finger
  - http
- Übersichtstabelle

Kapitel 35

EINFÜHRUNG

Dieser Teil soll sich mit den Standarddiensten im Internet beschäftigen. Diese Dienste gibt es bereit schon seit den Anfängen von TCP/IP. Die Installierung der Dienste wird hier primär auf UNIX Systemen gezeigt, andere Betriebssysteme halten sich dort etwas zurück. Beispiele wird es im SuSE-Linux, Solaris 2.6 und Solaris 7 geben, sofern es keine Unterschiede zwischen den UNIX Varianten gibt steht UNIX da.

Die Dienste werden hier nach Portnummern sortiert aufgelistet und beschrieben.

Kapitel 36

STANDARDPORTS

Die Standarddienste benutzen normalerweise auch standariesierte Portnummern, siehe [RFC 1340][1] dort stehen einige. Die Konfigurationsdatei der Ports ist die /etc/services. Ist in dieser Datei eine Portnummer nicht registriert, dann kann auch der Dienst nicht gestartet werden. Das gilt auch für selbst geschriebene Client/Server Applicationen. Die Portnummer kleiner 1023 sind standariesiert und priviligiert. Ports mit groesseren Nummer können je nach bedarf verteilt werden.

/etc/services

Die /etc/services ist auf jedem TCP/IP fähigen Betriebssystem installiert. Der Ort dieser Datei ist jedoch unterschiedlich :

Betriebssystem	Standort
UNIX	/etc/services
NOVELL	SYSTEM:ETC/SERVICES
OS/2 WARP SERVER 4.0	C:/MTP/ETC/SERVICES
WINDOWS NT	C:/WINNT/DRIVER/ETC/SERVICES
WINDOWS 95	C:/WINDOWS/SERVICES

Tabelle 36.1: Standorte der /etc/services

- Die /etc/services ist eine ASCII Text Datei
- Verfügbare Manpages : services
- Jede Zeile $\equiv$ einem Porteintrag
- Kommentare werden durch ein Hash eingeleitet und gehen bis zum Ende einer Zeile
- Jede Zeile besitzt mindestens zwei Spalten, wobei die Spalten entweder durch ein Leerzeichen oder durch ein Tab getrennt werden
- Die Bedeutung der Spalten ist aus Tabelle 36.2 zu entnehmen

Tabelle 36.2: Spalteninhalte der /etc/services

Feld	**Beschreibung**
1	Symbolischer Name der Portnummer
2	Portnummer und Transport-Protocol
3..	Optionale Aliasnamen

- Beispieleinträge könnten so aussehen :

[1] RFC-1340 Assigned Numbers

Quelltext 36.0.1 Beispiel einiger Serviceeinträge in der /etc/services

```
# /etc/services
ftp             21/tcp        # FTP ueber TCP Port 21
telnet          23/tcp        # Telnet
smtp            25/tcp   mail # EMAIL ueber smtp oder mail ueber TCP Port 25
syslog          514/udp       # Systemlog ueber UDP Port 514
```

Kapitel 37

INETD

Das `inetd` ist ein Programm der es ermöglicht mehrere Dienste zentral zu verwalten. Sei es in der Verfügbarkeit oder in den Sicherheitseinstellungen der einzelnen Dienste. Der `inetd` (Internet Daemon) wird hauptsächlich auf BSD Systemen verwendet. Unter SYSTEM V Systemen werden die einzelnen Dienste durch eigene Programme abgebildet. Solaris als System V System bietet den inetd jedoch auch.

Wir werden uns den `inetd` mal genauer anschaun.

37.1 Funktionsweise

Wenn der `inetd` gestartet wird, liest er seine Konfigurationsdatei, die /etc/inetd.conf (siehe Abschnitt 37.2), ein und bewertet diese. In der Konfigurationsdatei stehen nun die Ports und Programme drinn. Der `inetd` hängt sich nun auf alle in der Konfigurationsdatei definierten Ports und wartet nun auf irgendein Connect. Wenn ein Client nun eine Verbindung zum Server aufbauen will, bekommt der `inetd` dieses mit, weil er sich an den Port gehängt hat. In der Konfigurationsdatei steht nun welches Programm er wie starten soll. Nachdem sich der der `inetd` durch ein `fork()` dupliziert hat, verbindet er den Eingabekanal aus dem Netzwerk mit dem Eingabekanal (stdin) des geforkten Prozesses, und der Ausgabekanal aus dem Netzwerk wird mit dem Ausgabekanal (stdout) des geforkten Prozesses verbunden. Danach startet der geforkte Prozess durch ein `exec()` das eigendliche Serverprogramm, welches in der Konfigurationsdatei steht. Für `inetd` ist die Verarbeitung beendet. Wenn das Serverprogramm beendet wird, wird automatisch der Netzwerkkanal geschlossen.

37.2 Die Konfigurationsdatei /etc/inetd.conf

1. Es ist eine ASCII-Textdatei
2. Jede Zeile $\equiv$ einer Port $\rightarrow$ Programmzuweisung
3. Ein Kommentar wird mit dem Hash eingeleitet und gilt bis zum Ende der Zeile
4. Eine Zeile wird in Spalten aufgeteilt, wobei der Spaltentrenner entweder Leerzeichen oder Tabs sind.
5. Die Spalten haben folgende Bedeutungen :

Tabelle 37.1: Spalteninhalte der /etc/inetd.conf

Feld	Beschreibung
1	Servicename, optional kann die Version durch ein Slash mit angegeben werden. z.B. rstatd/2-4 soll sagen der Dienst rstat von Version 2 bis 4

▼

▲

2	Sockettyp der Verbindung **stream** Streamübertragungen für TCP **dgram** Datagrammübertragungen für UDP **raw** für Rohdaten **seqpacket** für hinereinanderfolgende Pakete (Solaris) **tli** für TLI (Solaris)
3	Protocol, das protocol sollte in der /etc/protocols stehen. **tcp** Streamübertragungen für TCP **udp** Datagrammübertragungen für UDP **rpc** RPC Dienstangebot
4	Die Startart gibt an wie der Dienst gestartet werden soll **wait** der `inetd` wartet auf das Ende des Dienstes **nowait** startet für jede Verbindung den Dienst ohne zu warten
5	Username unter dem das Programm gestartet werden soll
6	Serverprogramm das gestartet werden soll
7	Argument 0 an Serverprogramm
8..	Weitere Argumente an das Serverprogramm

37.3 Beispielkonfiguration

Mal angenommen wir wollen einen Dienst implementieren, z.B. einen Telnet-Server. Dann trägt man zuerst alle Informationen zusammen. Telnet benötigt TCP als Übertragungsprotocol, Telnet selbst läuft auf Port 23 (telnet) und arbeitet mit Streams. Damit sich mehrere Benutzer gleichzeitig einloggen können darf der `inetd` nicht auf die Beendigung warten. Und der Server heisst `in.telnetd`. Der Server selbst muss mit *root*-Rechten laufen, weil er die /etc/shadow lesen muss um das Passwort überprüfen zu können. Woweit so gut, damit kann man nun die Konfigurationszeile zusammensetzen (siehe Quelltext 37.3.1)

Quelltext 37.3.1 Beispieleintrag in der /etc/inetd.conf

```
telnet stream tcp nowait root /usr/sbin/in.telnetd in.telnetd
```

Kapitel 38

ÜBERSICHT DER EINZELNEN DIENSTE

38.1 echo

Der Dienst *echo* macht eigendlich nichts anderes als die Zeichen die ankommen wieder zurück zu senden. Dieser Dienst wird intern vom `inetd` bereitgestellt und dient zu Testzwecken. Nach dem Test sollte man diesen ausschalten.

Tabelle 38.1: Eigenschaften des Services *echo*

Servicename	Portnummer	TCP	UDP	Aliasnamen
echo	7	✓	✓	-

Server Wird intern von `inetd` bereit gestellt. Um den Dienst zu aktivieren/deaktivieren muß der Eintrag in der /etc/inetd.conf entsprechend angepasst werden.

Client Als Client kann z.B. das Programm `telnet` verwendet werden. Jedoch Vorsicht ! Telnet lässt sich dann nur noch extern über ein `kill` beenden, da auch ein CTRL-C ge'echo't wird.

Aufruf : `telnet <hostname> echo` oder `telnet <hostname> 7`

38.2 discard

Der Dienst *discard* macht eigendlich nichts anderes als die Zeichen die ankommen zu ignorieren.

Tabelle 38.2: Eigenschaften des Services *discard*

Servicename	Portnummer	TCP	UDP	Aliasnamen
discard	9	✓	✓	sink, null

Server siehe *echo*

Client Als Client kann z.B. das Programm `telnet` verwendet werden. Jedoch Vorsicht ! Telnet lässt sich dann nur noch extern über ein `kill` beenden, da auch ein CTRL-C ignoriert wird.

Aufruf : `telnet <hostname> discard`

38.3 systat

Der Dienst *systat* hat kein definiertes Ausgabeformat. Im Normalfall jedoch gibt *systat* eine komplette Prozessliste des Hosts zurück.

Tabelle 38.3: Eigenschaften des Services *systat*

Servicename	Portnummer	TCP	UDP	Aliasnamen
systat	11	✓	-	-

Server Als Serverdienst arbeitet hier meist das Programm `ps`. Unter BSD Systemen wird ein `ps ax` unter SV ein `ps -e` ausgeführt. Die Ausgabe das Programms wird über das Netzwerk geleitet und kommt so am Client an.

Client Als Client kann z.B. das Programm `telnet` verwendet werden. Die Verbindung wird vom Server automatisch beendet.
Aufruf : `telnet <hostname> systat`

38.4 daytime

Der Dienst *daytime* gibt die eingestellte Systemzeit des Remotehost wieder zurueck.

Tabelle 38.4: Eigenschaften des Services *daytime*

Servicename	Portnummer	TCP	UDP	Aliasnamen
daytime	13	✓	✓	-

Server Der Dienst wird vom `inetd` selbst gestellt. Er gibt die Systemzeit im Format von `date` aus.

Client Als Client kann z.B. das Programm `telnet` verwendet werden.
Aufruf : `telnet <hostname> daytime`

38.5 netstat

Der Dienst *netstat* hat auch kein definiertes Ausgabeformat. In den meistenfällen wird das orginalausgabeformat des Programms `netstat` verwendet.

Tabelle 38.5: Eigenschaften des Services *netstat*

Servicename	Portnummer	TCP	UDP	Aliasnamen
netstat	15	✓	-	-

Server Als Server kommt hier meist das Programm `netstat` zum Einsatz der alle aktiven Verbindungen ausgibt und dem Client übermittelt.

Client Als Client kann z.B. das Programm `telnet` verwendet werden.
Aufruf : `telnet <hostname> netstat`

38.6 ftp

Der Dienst *ftp*. FTP benutzt immer zwei Ports ! Der Port 21 wird zum Verbindungsaufbau und zur Übermittlung von FTP-Befehlen benutzt. Der Port 20 hingegen wird genutzt wenn Daten übertragen werden. Es folgt nun ein kleine Anriß, auf den FTP Server gehen wir später noch ein.

Mit FTP können Dateien von einem Host zum anderen bewegt werden. Download/Upload sozusagen.

Tabelle 38.6: Eigenschaften des Services *ftp*

Servicename	Portnummer	TCP	UDP	Aliasnamen
ftp-data	20	✓	-	-
ftp	21	✓	-	-

Server Es gibt zur Zeit 2 Server die sehr verbreitet eingesetzt werden.

- `in.ftpd` ist der kleinste FTP Server. Den gibts überall
- `in.wuftpd` sehr komplexer Server mit großer Konfigurationsmöglichkeit was Rechte und so angeht

Der `in.wuftpd` ist unter Solaris nicht standardmäßig dabei und muß sich besorgt werden.

Client Als Client bietet jedes Betriebssystem FTP-Clients an. Unter UNIX meist mit dem Namen `ftp`. Unter Windows ist das Bordmittel, welches von MS geliefert wird so schlecht, daß es tausend andere FTP-Clients gibt.

38.7 telnet

Der Dienst *telnet* ermöglicht die Remotekontrolle eines Hosts. Dabei wird einem die Console über das Netzwerk zur Verfügung gestellt. Die localen Resourcen werden nicht genutzt, können auch nicht.

Tabelle 38.7: Eigenschaften des Services *telnet*

Servicename	Portnummer	TCP	UDP	Aliasnamen
telnet	23	✓	-	-

Server Der Server ist in den meisten Fällen der `in.telnetd` auf den wir noch eingehen werden

Client Als Client Programm kommt unter UNIX `telnet` in Frage. Bei MS gilt das welches bereits bei *ftp* angesprochen wurde.

38.8 smtp

Der Dienst *smtp* ermöglicht den Austausch von EMails im Netzwerk.

Tabelle 38.8: Eigenschaften des Services *smtp*

Servicename	Portnummer	TCP	UDP	Aliasnamen
smtp	25	✓	-	-

Server Als Server wird unter UNIX meist `sendmail` benutzt. Zu dem es demnächst in diesem Kino ein extra Kapitel geben wird. Unter Windows muß man sowas kaufen, nennt sich dann meist Exchange oder so alternativ kann dort aber auch sendmail verwendet werden.

Client Als Client kommt unter UNIX das Programm `mail` zum Einsatz, welches locale und Netzmail verwalten kann. Jedoch wird meist ein Grafisches Tool bevorzugt oder `pine` der ultimative EMail Client der auch über telnet sehr gut funktioniert. Windows ?!? ähm.... ja !

38.9 domain

Der Service *domain* wird für Nameserver anfragen benötigt. Ein Nameserver ist ein zentraler Dienst der Hostnamen zu IP Addressen umwandelt.

Tabelle 38.9: Eigenschaften des Services *domain*

Servicename	Portnummer	TCP	UDP	Aliasnamen
domain	53	✓	✓	nameserver

Server Als Server wird unter UNIX meist die BIND Implementation genutzt. Unter Windows NT implementiert MS eine abart des BIND's. Der Server verwaltet nun die entsprechenden Listen und gibt diese auf Anfrage preis.

Client Als Client kommt das Betriebssystem selbt in Frage, Das Betriebssystem besitzt eine Funktion mit dem Namen `gethostbyname()` die zu erst local sucht danach den Nameserver fragt. Die Funktion gibt dann entsprechend dir IP zurück.

38.10 bootp

Die BOOTP Dienste werden benötigt um IP Addressen einem Host zuzuweisen. Das kann dynamisch passieren oder Statisch. Meist im Einsatz bei Diskless-Stations.

Tabelle 38.10: Eigenschaften des Services *bootp*

Servicename	Portnummer	TCP	UDP	Aliasnamen
bootps	67	✓	✓	-
bootpc	68	✓	✓	-

Server Der Server benutzt *bootps* und besitzt eine MAC-¿IP Tabelle. Bei Anfragen gibt er die IP wieder zurück.

Client Der Client benutzt *bootpc* und sendet eine Anfrage mit seiner MAC Addresse ins Netzwerk. Er sollte dann seine IP vom Server bekommen.

38.11 tftp

Der *tftp* implementiert die simpelste Art eines FTP Servers. Das tftp[1] wird meist in Verbindung mit Diskless-Stations benutzt. Die DLS bezieht über tftp den Kernel der dann von der DLS gestartet wird.

Tabelle 38.11: Eigenschaften des Services *tftp*

Servicename	Portnummer	TCP	UDP	Aliasnamen
tftp	69	-	✓	-

Server Unter UNIX wird dort meist `in.tftpd` gestartet. Man sollte dort auf jeden Fall die Sicherheitseinstellungen überprüfen.

Client Unter UNIX gibts das Tool `tftp` mit dem man den tftp-Server testen kann.

38.12 gopher

Gopher war der erste Dienst im Internet der ein Verzeichnissdienst auf graphischer Ebene ermöglicht hat. Durch einfache Items konnte man so in der Verzeichnisstruktur rumwandern. Gopher wurde dann durch das World Wide Wait ersetzt. Einige Wetterstationen bieten immernoch mit sehr grossen Andrang Gopher an, da es einfacher und schneller ist.

Tabelle 38.12: Eigenschaften des Services *gopher*

Servicename	Portnummer	TCP	UDP	Aliasnamen
gopher	70	✓	✓	-

Server Ja, den gibt es unter Linux leider nicht mehr, ohne diesen selbst zu implementieren. Unter Windows NT wird ein Gopher mit dem IIS geliefert.

Client Das Programm `gopher` dient als Client unter UNIX. Aber auch Webbrowser können mit Gopher umgehen.

38.13 finger

[1] `tftp`-Trivial File Transfer Protocol

Tabelle 38.13: Eigenschaften des Services *finger*

Servicename	Portnummer	TCP	UDP	Aliasnamen
finger	79	✓	-	-

Server

Client

38.14 http

Das http[2] dient zum Austausch von Daten. Ähnlich wie FTP und Gopher bietet HTTP auch einen Verzeichnissdienst an. Der meistgenutzte Verzeichnissdienst ist wohl im World Wide Wait zu finden. HTTP hat den Gopher abgelöst.

Tabelle 38.14: Eigenschaften des Services *http*

Servicename	Portnummer	TCP	UDP	Aliasnamen
http	80	✓	✓	www

Server Als Server wird meist der Apache Webserver eingesetzt. Aber auch Netscape Server, oder MSIIS, etc.

Client Als Client kann man die handelsüblichen Browser benutzen, wie Netscape, IE, Mozilla, etc. Aber auch nicht grafische wie lynx, der sich hervorragend für Bedienungsanleitungen etc eignet.

[2] http-Hypertext Transferprotocol

Kapitel 39

ÜBERSICHTSTABELLE

Tabelle 39.1: Eigenschaften des Services *Alle Dienste*

Servicename	**Portnummer**	**TCP**	**UDP**	**Aliasnamen**
bootpc	68	✓	✓	-
bootps	67	✓	✓	-
daytime	13	✓	✓	-
discard	9	✓	✓	sink, null
domain	53	✓	✓	nameserver
echo	7	✓	✓	-
finger	79	✓	-	-
ftp-data	20	✓	-	-
ftp	21	✓	-	-
gopher	70	✓	✓	-
http	80	✓	✓	www
netstat	15	✓	-	-
smtp	25	✓	-	-
systat	11	✓	-	-
telnet	23	✓	-	-
tftp	69	-	✓	-

TEIL IX

NETWORK FILESYSTEM

- Einführung
  - Funktionsprinzip
  - Übersicht der Programme und Konfigurationsdateien
- Allgemeine Implementation
  - Portmapper
  - NFS-Server
  - NFS-Client
  - NFS Diagnose Tools
- Linux NFS Implementation
  - Portmapper
  - NFS-Server
- Solaris NFS Implementation
- Praktikum (Linux-SuSE)
  - First Try
  - The ID Puzzle

Kapitel 40

EINFÜHRUNG

Das NFS[1] wird von UNIX basierenden Betriebssystemen benutzt, um Festplattenkapazitäten über das Netzwerk zur Verfügung zu stellen. Das NFS wurde von Sun Microsystems entwickelt und hat sich zum Standard unter UNIX entabliert. Zur Zeit gibt es zwei noch primär genutzte Versionen. Zum einen die Version 2 die nur UDP unterstützt. Die neuere Version 3 unterstützt TCP und Serverübernahmen (Failover).

40.1 Funktionsprinzip

Das NFS bietet einige einfache Befehle zur Komunikation. Zum Beispiel der Befehl *lese n Bytes* oder *lösche Datei xy*. Der Client sendet den Befehl zum Server. Dieser gibt dann das Resultat des Befehls bekannt, sprich die angeforderten Bytes oder ein OK, je nach Befehl.

40.1.1 Befehle und Kommunikation

Man unterscheidet zwischen den Befehlen zwischen *wiederholbaren* und *nicht wiederholbaren* Befehlen.

Wiederholbare Befehle sind z.B. das Lesen oder Schreiben von Daten in eine Datei, passiert der Vorgang auf dem Server zweimal, dann hat daß keine Auswirkungen auf die Datei selbst.

Nicht wiederholbare Befehle sind Befehle die bei der Wiederholung einen Fehler bringen würden. Eine Datei zu löschen zum Beispiel wäre ein solcher Befehl. Würde der zweimal kommen, würde der erste Befehl erfolgreich ausgeführt werden, der zweite jedoch bringt eine Fehlermeldung, weil die Datei ist ja nicht mehr da.

Das Problem der Unterscheidung ergibt sich, da NFS Version 2 über UDP abgehandelt wird. UDP bietet keinerlei Leitungssicherheit. Die Sicherung der Daten muß die Anwendung übernehmen. Es kann jedoch vorkommen das die Antwort eines Servers unterwegt verloren geht. In diesem Fall sendet der Client den Befehl nochmal (wie es TCP auch tun würde), und genau jetzt kommt der gleiche Befehl zweimal zum Server. Würde der Sever nun mit einer Fehlermeldung reagieren und diese Antwort würde zum Client durchkommen, würde der Anwender plötzlich eine Fehlermeldung bekommen, obwohl die Datei gelöscht wurde.

Um das zu verhindern behält der NFS-Server einen Cache bereit, wo er alle nicht wiederholbaren Befehl speichert und diesen Cache mit den Befehlen vergleicht die ein Client gerade absendet. Sollte sich im Cache der gleiche Befehl befinden wird einfach nur der letzte Status zurückgesendet und im Cache wird dem Eintrag ein Refresh verpasst. Nach einer gewissen Zeit verfällt der Eintrag im Cache. Die Zeitdauer wird vom Server dynamisch, jenach Leitungsqualität und Geschwindigkeit angepasst.

[1] NFS-Network Filesystem

In der Version 3 ist dieser Mechanismus nicht mehr aktiv. Es kann zwar auch vorkommen das ein ACK das Ziel nicht erreicht, und der Client sendet es automatisch nochmal, jedoch verwirft der TCPIP-Stack das Segment gleich wieder, da er anhand der Sequenznummer sieht, daß er dieses Segment bereits schon hatte.

40.1.2 Aufgaben eines NFS-Servers

Die Aufgabe eines NFS-Servers ist die Bereitstellung von Festplattenkapazitäten. Dem NFS-Server wird durch eine Konfigurationsdatei mitgeteilt welche Verzeichnisse er an welche Host exportieren darf. Das Verzeichniss muß auf dem NFS-Server lokal vorhanden sein. Durch eine Einstellung ist es möglich, daß ein NFS-Server entfernt eingehängte Verzeichnisse exportiert. Zum Beispiel wird ein Verzeichniss via Samba von einem NT-Server gemountet, welches der NFS-Server selbst als NFS wieder exportiert (Gateway).

40.1.3 Aufgaben eines NFS-Clients

Der NFS-Client benutzt nun die exportierten Verzeichnisse eines NFS-Servers und mountet diese selbst irgendwo im eigenen Dateibaum ein. Dabei muß der Verzeichnissname auf dem Server und dem Client nicht unbedingt der gleiche sein. Es gelten die gleichen Spielregen wie beim Mouten von lokalen Resourcen, sprich der Mountpoint muß vorhanden sein, etc...

40.1.4 Voraussetzungen zum Betrieb

Für NFS wird der Portmapper benötigt. Dieser Dienst kann ein Port multiplexen. Soll heissen das eigentlich mehrere Ports durch einen Port gesendet werden. Der Portmapper hat die Aufgabe auf beiden Seiten nun die Daten wieder korrekt zu entmultiplexen.

Für nicht UNIX Systeme wird immer das entsprechende Programm zur Benutzung von NFS benötigt. Wobei es für OS/2 und BeOS freie Implementationen gibt die auch Funktionieren. Unter NT benötigt man ThirdParty Tools, wie etwa Reflections/X. Wie es auf Macs aussieht weiss ich nicht. Unter Novell Netware 4.11 ist NFS bei den FTP - Tools dabei, der NWAdmin erlaubt dort auch die Konfiguration von Rechten auf UNIX Basis.

Weiterhin, was glaube ich nicht zu erwähnen ist, sollte TCP/IP installiert sein.

40.1.5 Einsatzgebiete und Vorteile von NFS

Durch die durchdachte organisation des UNIX-Dateibaums und durch die Transparenz die UNIX einem Benutzer bietet ist es möglich ein Softwarepaket auf einem Server zu installieren und das Installationsverzeichniss entsprechent zu expotieren. Die Clients können dann das exportierte Verzeichniss in ein Verzeichniss gleichen Namens mounten. Dardurch ergibt sich auf dem Client der Eindruck, daß das Programm dort lokal läge. Das Programm kann nun auf dem Client genutzt werden. Wenn ganze Arbeitsgruppen von ca. 200 Clients das tun, ist die Software auf allen Clients lauffähig. Ist ein Update einzuspielen geschieht dieses nur am Server und die Clients verfügen ab diesem Zeitpunkt dann die neuere Version der Software.

Weiterhin kann mittels NFS der Platz auf lokalen Laufwerken eingespart werden, indem Teile die auf jeden Client und Server vorkommen auf dem Client gelöscht werden und vom Server gemountet werden. Das spart Administrationsarbeit für die Software. Da fast alle Softwareprodukte sich in einem Verzeichniss installieren indem sie in diesem Verzeichniss nochmal ein bin, lib, etc, Verzeichniss anlegen, wo das Binary und die Librarys sich befinden. Zum Vergleich: Unter Windows installiert sich

eine Software kreuz und quer über die Verzeichnisse und ersetzt sogar Betriebssystemlibrarys, teilweise laufen die Programme erst nach einem Neustart des Systems. Unter UNIX kommt so etwas nur mit Systemkomponenten vom Betriebssystem vor, externe Programme tun dieses nicht.

40.1.6 Nachteile beim Einsatz von NFS

Wenn NFS benutzt wird gelten die Informationen der INode vom Server, ausgewertet werden diese Informationen jedoch auf dem Client. Dieser relativ kurze Satz bringt mehrere Gefahren mit sich. Als da wäre z.B. die Systemzeit, die sollte auf dem Server und dem Client die gleiche sein, ansonsten kann es dazuführen das der Benutzer ständig eine Fehlermeldung bekommt das die Datei neuer ist als die die er gerade Speichern möchte.

Weiterhin muessen die UID und GID auf dem Server und dem Client übereinstimmen. Mal angenommen auf dem Server gibts es ein Administrator *otto* der hat die UID 400. Auf dem Client existiert ein Benutzer *karl* mit der UID 400. Wenn der Benutzer *karl* nun auf eine Datei zugreift liest der Client die INode aus und vergleicht die UID mit dem eingeloggten Benutzer. Da $400 \equiv 400$ ist, wird der Zugriff gewährt. Der Admin steht dann Kopf. Das soll heissen das beim Einsatz von NFS die /etc/passwd und /etc/group synchonisiert werden muß. Dazu bietet sich dann NIS an.

40.1.7 Position im ISO-OSI Modell

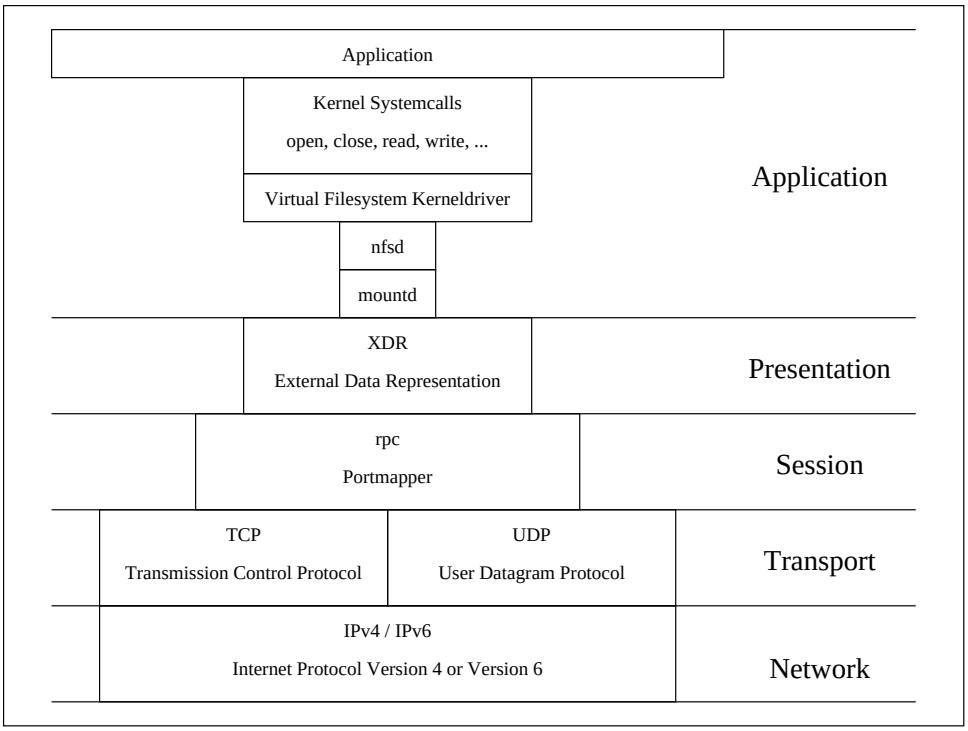

Abbildung 40.1: NFS Position im ISO-OSI Modell

40.2 Übersicht der Programme und Konfigurationsdateien

Tabelle 40.1: Programme und Dateien in der NFS Umgebung

Programm / Datei	**Bedeutung**
`rpc.mountd`	Mount Deamon
`rpc.nfsd`	NFS Daemon
`rpc.statd`	Status Daemon
`rpc.lockd`	Lock Daemon
`rpc.quotad`	Quota Daemon
`showmount`	Anzeigen von Exporten
`rpcinfo`	RPC Informationen
`mount`	Mounten von Devices
`/etc/exports`	Liste der zu exportierenden Verzeichnisse
`/etc/fstab`	Mountings beim Booten
`/etc/rpc`	Konfiguration von RPC
`rcrpc`	Startmodul zum Portmapper
`rcnfsserver`	Startmodul für den NFS Server
`rcnfs`	Startmodul für NFS Client

Kapitel 41

ALLGEMEINE IMPLEMENTATION

Dieses Kapitel soll die allgemeine Implementation für alle UNIX Varianten abdecken. Die spezialitäten der einzelnen Derivate sollen im folge Kapitel beschrieben werden. Sofern nichts anderes dort steht bezieht sich der Text auf alle Linux Varianten, Solaris 2.6 und Solaris 7 Versionen.

41.1 Portmapper

Der Portmapper wird auf dem NFS-Server und dem NFS-Client benötigt. Der Portmapper überwacht den Port 111/tcp und 111/udp und vergibt dem anfragenden Host eine neue Portnummer für eine Verbindung. Diese Eigenschaft wird genutzt um den Port-Pool von TCP und UDP zu erweitern.

Damit der Portmapper auch funktioniert muß der Port 111 in der /etc/services entsprechend eingetragen sein :

Quelltext 41.1.1 Für den Portmapper benötigte Einträge in der /etc/services

```
sunrpc          111/udp         rpcbind
sunrpc          111/tcp         rpcbind
```

Diese Einträge sind in den meisten Fällen bereits drin. Danach muß der Portmapper selbst noch konfiguriert werden. Der Portmapper benutzt dazu die /etc/rpc. In dieser Datei stehen die erweiterten Ports zu den Programmen drin. Um NFS zu ermöglichen müssen die folgenden Einträge vorhanden sein, meist auch schon drin :

Quelltext 41.1.2 Für NFS benötigte Portmapperkonfiguration in der /etc/rpc

```
portmapper      100000  portmap sunrpc rpcbind
nfs             100003  nfsprog
mountd          100005  mount showmount
rquotad         100011  rquotaprog quota rquota
nlockmgr        100021
status          100024
```

Danach muß der Portmapper gestartet werden, zuvor sollte man Prüfen ob dieser nicht bereits läuft, wenn ja dann einfach killen und neustarten. Nachdem der Portmapper nun läuft kann der NFS-Server konfiguriert werden.

41.2 NFS-Server

Der NFS-Server selbst besteht aus mehreren Prozessen.

41.2.1 Mount Daemon

Der *mountd* der als erstes gestartet werden sollte, kontrolliert die Zugriffsberechtigungen auf Hostebene. Nachdem die Autorisierung des Hosts abgeschlossen wurde vergibt der *mountd* eine Verbindungsnummer und stellt die Verbindung zum *nfs* her.

41.2.2 NFS Daemon

Der *nfs* behandelt die vom Client kommenden Befehle. Er hält auch die Liste der nicht wiederholbaren Befehle. Er prüft die Rechte auf Benutzerebene ebenfalls.

41.2.3 Lock Daemon

Der *nlockmgr* kümmert sich um Netzwerk Dateisperren. Der Daemon hat die Aufgabe das Filelocking über das Netzwerk zu implementieren. **WARNUNG: Wird von Linux NICHT unterstützt**

41.2.4 Status Daemon

Der *status* gibt den Filelock Status einer Datei bekannt. Er sollte immer mit dem *nlockmgr* zusammen. **WARNUNG: Wird von Linux NICHT unterstützt**

41.2.5 Quota Daemon

Der *quotad* überwacht auf exportierten Verzeichnissen den Quota eines Benutzers. **WARNUNG: Wird von Linux erst ab Kernel 2.2.x unterstützt**

41.2.6 /etc/exports

`/etc/exports`

Die `/etc/exports` legt fest welche Verzeichnisse an welche Clients exportiert werden sollen.

- Die `/etc/exports` ist eine ASCII Text Datei
- Verfügbare Manpages : `exports(5)`
- Jede Zeile ≡ ein exportierendes Verzeichnis
- Kommentare werden durch ein Hash eingeleitet und gehen bis zum Ende einer Zeile
- Jede Zeile besitzt genau zwei Spalten, wobei die Spalten entweder durch Leerzeichen oder durch Tab's getrennt werden
- Die Bedeutung der Spalten :

Tabelle 41.1: Spalteninhalte der `/etc/exports`

Feld	Beschreibung
1	Verzeichnis

2	Berechtigung. Wobei hier der Hostname gefolgt von den Berechtigungen in Klammern notiert wird. Wird der Hostnamen weggelassen findet keine Prüfung statt, und jeder Client darf. Format : *hostname*(*access*),...

- Als Zugriffsflags (access) können folgende in Frage kommen :

Access Flag	Bedeutung
`ro`	Readonly
`rw`	Readwrite
`root_squash`	Die UID 0 wird beim Zugriff in *nobody* gemappt
`no_root_squash`	Auf fremde *root*'s haben Rootrechte
`squash_uids`	Liste von UID's die beim Zugriff auf *nobody* gemappt werden sollen. Beispiel: `squash\_uids=0-10,20,30,40-50`
`squash_gids`	das gleiche für Gruppen

Tabelle 41.2: Zugriffsflags der /etc/exports

- Beispieleinträge könnten so aussehen :

Quelltext 41.2.1 Beispiel einiger Exportseinträge in der /etc/exports

```
# /etc/exports
/home *.hurst.pnet(rw,root_squash)
/usr/lib/java puma.hurst.pnet(ro),falcon.hurst.pnet(rw,no_noot_squash)
/var/spool tiger.hurst.pnet(rw,squash_uids=0-100)
```

Hier wird /home an alle Hosts der Domain HURST.PNET freigegeben. Das Java Verzeichnis wird nur dem PUMA und FALCON freigegeben. Und /var/spool wird an TIGER freigegeben, wobei Zugriffe von Benutzern unter UID 100 als *nobody* gemappt werden.

41.3 NFS-Client

Der NFS-Client benötigt, wie der Server auch, den Portmapper. Zum Mounten von NFS Exporten jedoch ist nur noch das Programm `mount` notwendig. Da `mount` auch auf die /etc/fstab bzw. /etc/vfstab zugreift, können NFS Mountings in der Datei eingetragen werden und beim Systemstart z.B. gemountet werden.

41.4 NFS Diagnose Tools

Linux selbst besitzt nicht die Welt an Diagnosetools. Genaugenommen eigentlich nur 2 ...

41.4.1 Portmapper Check

Um den Portmapper zu überprüfen kann das Programm `rpcinfo` verwendet. Das Programm gibt die RPC-Port aus die zur Zeit belegt sind. Das Programm besitzt folgende Syntax :

```
rpcinfo -p [hostname]
```

Unter Solaris ist eine erheblich erweiterte Informationsausgabe verfügbar. Unter Linux werden die RPC-Ports des Hosts *hostname* ausgegeben :

```
crow: dozent as root # rpcinfo -p tiger
   program vers proto   port  service
    100000    2   tcp    111  rpcbind
    100000    2   udp    111  rpcbind
    100024    1   udp    698  status
    100024    1   tcp    700  status
    100005    1   udp    783  mountd
    100005    2   udp    783  mountd
    100005    1   tcp    786  mountd
    100005    2   tcp    786  mountd
    100003    2   udp   2049  nfs
    100003    2   tcp   2049  nfs
crow: dozent as root #
```

Bildschirmausschnitt 41.4.1: Portmappertest mittels `rpcinfo`

Damit weiss man nun das auf dem TIGER ein NFS Server läuft.

41.4.2 Anzeige von Remote NFS Mountings

Das Programm `showmount` listet die aktuellen oder alle möglichen exportierten Verzeichnisse eines NFS-Servers an. Das Programm kann wie folgt aufgerufen werden :

```
showmount -a [hostname]
showmount -d [hostname]
showmount -e [hostname]
```

Wenn *hostname* nicht angegeben wird, wird immer der Lokalehost ausgefragt.

Mit der Option *-a* wird eine Liste ausgegeben, die alle aktuellen exportierten Verzeichnisse anzeigt. Und zwar nur die die wirklich auch gerade von NFS-Clients gemountet sind.

Die Option *-d* zeigt nur die Verzeichnisse ohne Hostnamen an.

Die Option *-e* zeigt alle möglichen Exporte mit den entsprechenden Hostberechtigungen an.

Dazu ein Beispiel mit zwei Hosts. Zum einen CROW und zum anderen `puma` :

```
crow: dozent as root # showmount -a crow
falcon.hurst.pnet:/
crow: dozent as root # showmount -d crow
/
crow: dozent as root # showmount -e crow
export list for crow:
/ (everyone)
crow: dozent as root # showmount -e puma
export list for puma:
/home *.hurst.pnet
/     *.hurst.pnet
crow: dozent as root # showmount -d puma
/
/home
crow: dozent as root # showmount -a puma
crow.hurst.pnet:/
crow.hurst.pnet:/home
eagle.hurst.pnet:/
falcon.hurst.pnet:/home
lx.hurst.pnet:/home
panther.hurst.pnet:/home
shark.hurst.pnet:/home
tiger.hurst.pnet:/home
crow: dozent as root #
```

Bildschirmausschnitt 41.4.2: Beispiele zu `showmount`

Aus dem Beispiel kann man erkennen das der Host `falcon` das Rootdirectory vom Host `crow` gemountet hat. Weiterhin ist zu erkennen das der Host `puma` das Rootdirectory sowie auch `/home` für alle Hosts aus der Domain HURST.PNET exportiert. Und als letztes bekommen wir eine Liste von den Hosts die von dem Export gebrauch machen.

Man kann also feststellen wer wo was mountet.

Kapitel 42

LINUX NFS IMPLEMENTATION

42.1 Portmapper

Der Portmapper nennt sich unter Linux `portmap`. Der Ort war früher das `/usr/sbin` welches im Verlauf der Zeit zu den eigendtlichen Ort bewegt wurde, nach `/sbin`.

Um den Portmapper zu stoppen oder zu starten kann die allgemeine Form verwendet werden in dem der Prozess direkt gekillt wird und neu gestartet wird.

42.1.1 Starten und Stoppen allgemein

```
tiger: dozent as root # ps ax|grep portmap
   91 ?        S      0:00 /sbin/portmap
  467 pts/0    S      0:00 grep portmap
tiger: dozent as root # kill -TERM 91
tiger: dozent as root # /sbin/portmap
tiger: dozent as root # ps ax|grep portmap
  488 ?        S      0:00 /sbin/portmap
  469 pts/0    S      0:00 grep portmap
tiger: dozent as root #
```

Bildschirmausschnitt 42.1.1: Stoppen und Starten des Portmappers UNIX like

42.1.2 Starten und Stoppen SuSE-Linux

Unter SuSE-Linux ist jedoch die Verwendung des entsprechenden Moduls ratsam :

```
/sbin/init.d/rpc {start|stop}
```

Tabelle 42.1: Optionen für das SuSE Modul `rpc`

Option	Beschreibung
`start`	Startet den Portmapper
`stop`	Stoppt den Portmapper

Der Anwendungsfall :

```
tiger: root # /sbin/init.d/rpc stop
Shutting down RPC services
tiger: root # /sbin/init.d/rpc start
Starting RPC portmap daemon
tiger: root #
```

Bildschirmausschnitt 42.1.2: Stoppen und Starten des SuSE-Portmaps

42.2 NFS-Server

42.2.1 Mount Daemon

Der *mountd* wird unter Linux mit dem Programm `rpc.mountd` bereit gestellt.

```
/usr/sbin/rpc.mountd [-f exportfile] [-r]
```

Tabelle 42.2: Parameter für `rpc.mountd`

Option	Beschreibung
`-f exportfile`	Angabe des Exportfiles. Ist der Parameter nicht angegeben, werden die exporte aus der /etc/exports gelesen
`-r`	Erlaubt den export von importieren Verzeichnissen. Wird als Gatewayfunktion benötigt

42.2.2 NFS Daemon

Der *nfs* wird unter Linux mit dem Programm `rpc.nfsd` bereit gestellt.

```
/usr/sbin/rpc.nfsd [-f exportfile] [-r] [numcopys]
```

Tabelle 42.3: Parameter für `rpc.nfsd`

Option	Beschreibung
`-f exportfile`	Angabe des Exportfiles. Ist der Parameter nicht angegeben, werden die exporte aus der /etc/exports gelesen
`-r`	Erlaubt den export von importieren Verzeichnissen. Wird als Gatewayfunktion benötigt
`numcopys`	Anzahl der *nfsd* Prozesse

42.2.3 Starten und Stoppen allgemein

Beim manuellen killen ist darauf zu achten das zuerst `rpc.nfsd` und danach `rpc.mountd` gekillt wird. Beim Starten jedoch anderst herum

```
falcon: root # ps ax | grep rpc
  100  ?  S    0:01 /usr/sbin/rpc.mountd
  103  ?  S  710:10 /usr/sbin/rpc.nfsd
18491  p1 S    0:00 grep rpc
falcon: root # kill -TERM 103
falcon: root # kill -TERM 100
falcon: root # /usr/sbin/rpc.mountd
falcon: root # /usr/sbin/rpc.nfsd
falcon: root # ps ax | grep rpc
18992  ?  S    0:00 /usr/sbin/rpc.mountd
18994  ?  S    0:00 /usr/sbin/rpc.nfsd
18493  p1 S    0:00 grep rpc
falcon: root #
```

Bildschirmausschnitt 42.2.1: Stoppen und Starten des NFS-Server

42.2.4 Starten und Stoppen SuSE-Linux

Wie so oben so auch hier. Unter SuSE sollte man das Modul entsprechend aufrufen :

```
/sbin/init.d/nfsserver {start|stop|restart|status|reload}
```

Tabelle 42.4: Optionen für das SuSE Modul `nfsserver`

Option	Beschreibung
`start`	Startet den NFS Server
`stop`	Stoppt den NFS Server
`restart`	Stoppt und Startet danach gleich wieder den NFS Server
`status`	Zeigt an ob der Server läuft oder nicht
`reload`	Veranlasst die Prozesse die Exportdatei neu einzulesen

Der Anwendungsfall :

```
tiger: root # /sbin/init.d/nfsserver stop
Shutting down NFS-Server
tiger: root # /sbin/init.d/nfsserver start
Starting NFS-Server
tiger: root # /sbin/init.d/nfsserver status
NFS Server is up
tiger: root #
```

Bildschirmausschnitt 42.2.2: Stoppen und Starten des SuSE-NFS-Servers

Kapitel 43

SOLARIS NFS IMPLEMENTATION

Kapitel 44

PRAKTIKUM (LINUX-SUSE)

44.1 First Try

1. Sammeln Sie sich zu zweier Gruppen. Einer von Ihnen ist Host A der andere Host B

2. Verlassen Sie die graphische Oberfläche

3. Geben Sie auf Host A das Verzeichniss `/usr/X11R6` zum Export frei. Nur Ihr nachbar sollte die Zugriffsberechtigung erhalten. Tragen Sie dazu die entsprechende Zeile in der `/etc/exports` ein. Exportieren Sie das Verzeichniss unbedingt **ReadOnly**.

4. Starten Sie den NFS Server auf Host A neu in dem Sie das SuSE Modul `/sbin/init.d/nfsserver` einmal mit *stop* und danach mit *start* aufrufen.

5. Weiter gehts auf Host B

6. Mounten Sie nun das exportierte Verzeichniss von Host A auf Host B nach `/mnt`. Prüfen Sie die funktionalität mittels `ls` im `/mnt` Verzeichniss.

7. Nachdem das Funktioniert hat, Unmounten Sie `/mnt` wieder. Wechseln Sie in das `/usr` Verzeichniss und benennen Sie das Verzeichniss `X11R6` nach `old.X11R6` um. Um Umbenennen können Sie `mv` verwenden.

8. Starten Sie nun die graphische Oberfläche mit `startx`, es sollte nicht mehr Funktionieren.

9. Erstellen Sie den Mountpoint `/usr/X11R6` mit `mkdir` und mounten das exportierte Verzeichniss nach `/usr/X11R6`.

10. Starten Sie danach die graphische Oberfläche. Jetzt sollte es wieder Funktionieren. Nur langsamer.

11. Versuchen Sie nun auf Host B eine Datei im `/usr/X11R6/bin` Verzeichniss zu löschen. Sollte selbst als *root* nicht funktionieren.

12. Der Praktische Nutzen ist nun das alles was mit der graphischen Oberfläche zu tun hat nur noch auf einem Host sich befindet. Wenn es zum Beispiel einen neuen X-Server für die Grafikkarten gibt wird einfach am Server das Binary ausgetauscht und fertig ist das Update. Oder wenn ein Programm neu installiert wird ist es auch gleichzeitig auf den Clients verfügbar.

13. Machen Sie die Änderungen wieder rückgängig.

44.2 The ID Puzzle

1. Gruppieren Sie sich wieder zu zweier Gruppen. Der eine spielt wieder Host A der andere Host B
2. Erstellen Sie auf Host A zwei Gruppen. Siehe Tabelle :

Gruppenname	GID
benutzer	1500
admins	1501

3. Erstellen Sie auf Host A die folgenden Benutzer. Für jeden Benutzer muß ein Heimatverzeichniss angelegt werden. Bei der Benutzung von `useradd` ist die Option *-m* anzugeben. Achten Sie auch auf die Gruppen :

Username	UID	GID	Homedir
manni	8001	benutzer	/home/manni
sabine	8002	benutzer	/home/sabine
frank	8003	admins	/home/frank

4. Prüfen Sie die Rechte mittels einem `ls -la /home`
5. Auf dem Host B erstellen Sie bitte die folgenden Gruppen :

Gruppenname	GID
verwalt	1500
gaeste	1501
benutzer	1502
admins	1503

6. Wie sollte es anderst sein. Erstellen Sie auf Host B auch die folgende Benutzer :

Username	UID	GID	Homedir
monika	8001	verwalt	/home/monika
public	8003	gaeste	/home/gaeste
sabine	8002	benutzer	/home/sabine
manni	8001	benutzer	/home/manni
frank	8003	admins	/home/frank

7. Auch hier prüfen Sie die Rechte
8. Geben Sie nun an beiden Hosts das /home zum Export frei. Starten Sie den NFS Server neu.
9. Erstellen Sie an beiden Hosts den Mountpoint /home1
10. Mounten Sie an beiden Hosts nun das /home Export vom Nachbarn nach /home1
11. Prüfen Sie nun die Zugriffsrechte der Verzeichnisse .... TILT ....
12. Der Administrator *frank* hat auf Host B kein Zugriff auf sein Heimatverzeichniss, jedoch der Benutzer *public* ploetzlich. Diese Katastrophe kann man nur umgehen wenn man die Benutzer auf globaler Ebene plant.
13. Überlegen Sie sich für beide Hosts eine einzige Ideale Userdatenbank mit Userid's und Groupid's. Verändern Sie danach die Userdatenbanken auf beiden Hosts entsprechend so, daß alles ohne Probleme funktioniert.

TEIL X

NETWORK INFORMATION SERVICE

- Einführung
  - NIS Domains
  - NIS Master Server
  - NIS Slave Server
  - NIS Client
  - NIS+
  - Datenbankdateien
  - Map Dateien
  - Übersetzung in Maps
  - Und grafisch siehts so aus
- UNIX Allgemein Konfiguration
  - Konfiguration der Abfragereihenfolgelabel cfgfile:nsswitch
  - Programme und Konfigurationsdateien
- Speziell Linux Allgemein Konfiguration
  - Konfiguration der Maps
- Speziell SuSE 6.x Linux Konfiguration
  - Pakete für NIS Installieren
  - Konfiguration des NIS Clients
  - Konfiguration des NIS Servers
  - Starten und Stoppen des NIS Servers
  - Konfiguration der Maps
  - Starten und Stoppen des NIS Clients
- Speziell Solaris Konfiguration

- Praktikum
  - First Try
  - Der NIS User
  - The Final

Kapitel 45

EINFÜHRUNG

Das NIS[1] wird meist in UNIX Umgebungen eingesetzt. Es hieß früher YP[2] wurde jedoch von der British Telekom eingeklagt. NIS wurde von Sun Micrisystems erfunden und entwickelt und gilt heute als Standard in UNIX Umgebungen.

Das NIS stellt im Netzwerk eine Datenbank bereit auf denen die Clients zugreifen können. Zu den gebräuchslichten Datenbanken gehören wohl die `passwd` und `group`.

Mit NIS wird eine Synchonisierung der verschiedenen Datenbanken erreicht. Ähnlich wie beim DNS Prinzip. Unter NIS kann z.B. die /etc/passwd im Netzwerk zur Verfügung gestellt werden, um eine einheitliche Benutzerdatenbank auf allen UNIX Rechnern zu ermöglichen. Damit wird dann auch das NFS ID-Puzzle gelöst.

NIS kann jedoch noch mehr, es kann z.B. auch die /etc/services oder die /etc/hosts zur Verfügung stellen. Das Anwendungsgebiet von NIS ersteckt sich darüber hinaus auch auf selbst Implementierte Datenbankdateien, wie z.B. ein Addressenkartei oder diversen Konfigurationsdateien.

45.1 NIS Domains

Eine NIS Domain ist ein logischer Zusammenschluss von n Rechnern die dieser Domain mit angehören. In einer NIS Domain werden die Datenbankdateien entsprechend verwaltet. Jeder Host kann dieser Domain beitreten, tut er dieses bekommt er die Datenbankdateien von dieser Domain. In anderen Domains existieren wiederum andere Datenbankdateien.

Jede Domain identifiziert sich durch einen Namen. Dieser Name sollte 8 Zeichen nicht überschreiten, aus Kompatiblietätsgründen.

Eine NIS Domain hat nichts mit dem physikalischen Standort oder einer DNS Domain was am Hut.

45.2 NIS Master Server

Ein NIS Master Server stellt für eine NIS Domain die nötigen Datenbankdateien zur Verfügung. Der NIS Masterserver sollte selbst Client der NIS Domain sein um Konfigurationsprobleme zu vermeiden.

45.3 NIS Slave Server

Ein NIS Slave Server kopiert sich die Datenbankdateien eines NIS Masterservers und stellt diese auch zur Verfügung. Der NIS Slave hat jedoch kein Recht die Datenbankdateien zu modifizieren. Ein NIS Slave macht in grossen Netzwerken sinn um den NIS Master zu entlasten. Weiterhin wird der NIS Slave dann wichtig wenn mal der NIS Master ausfällt. Wieviele NIS Slaveserver man aufsetzt ist unrelevant und wird meist durch die größe des Netzwerkes entschieden.

[1] `NIS`-Network Information Services

[2] `YP`-Yellow Pages

45.4 NIS Client

Der NIS Client benutzt den NIS Master oder NIS Slave für diverse Auflösungen. z.B. Benutzer-ID oder Gruppen-ID oder Hostnamen, oder Serviceseinträge, etc. Die meisten UNIX Derivate verfügen eine Datei mit dem Namen /etc/nsswitch.conf die die jeweilige Abfrage Reihenfolge des Betriebssystems festlegt.

45.5 NIS+

Das NIS+ arbeitet wie NIS. Jedoch benutzt NIS+ ein Authorisierungmechanismus und ist für sehr große Netzwerke besser geeignet. Weiterhin wird bei NIS+ das Protocol TCP genutzt. Weiterhin ist der Aufbau von Baumartigen NIS+ Domains möglich, ähnlich wie es DNS tut.

45.6 Datenbankdateien

Die Datenbankdateien die ein NIS Server führt werden nach außen hin als *Maps* geführt. Jede Datenbankdatei besitzt meist zwei Maps. Eine ist sortiert nach Namen, die andere Sortiert nach ID's. Diese Maps haben auch einen ähnlichen Namen wie die Datenbankdateien selbst. Die folgende Tabelle soll die verschiedenen Maps auflisten :

Dateiname	Mapname	Bedeutung
passwd	passwd.byname , passwd.byuid	Benutzerverwaltung
group	group.byname , group.bygid	Gruppenverwaltung
hosts	hosts.byname, hosts.byaddr	Hostnamenverwaltung
ethers	ethers.byname, ethers.byaddr	Ethernet MAC Addressen
networks	networks.byname, .byaddr	Netzwerkverwaltung
rpc	rpc.bynumber	RPC Diensteverwaltung
services	services.byname, .byservices	Verwaltung von Portnummern
protocols	protocol.byname, .bynumber	Netzwerkprotokolle
netgroups	netgroup.byhost, .byuid, .bynet	NIS Gruppen/Host/User Verwaltung
bootparams	bootparams	NFS Bootmounts Verwaltung
aliases	mail.aliases, mail.byaddr	Mailalias Verwaltung
publickey	publickey.byname	Passwörter für Secure RPC
netid	netid.byname	Benutzer/Host Verwaltung für S-RPC
passwd.adjunct	passwd.adjuct.byname	Benutzer für C2 Sicherheit
group.adjunct	group.adjuct.byname	Gruppen für C2 Sicherheit
netmask	netmask.byaddr	Verwaltung der Netzwerkmasken
timezone	timezone.byname	Verwaltung der NIS Zeitzone
auto.master	auto.master	Konfiguration für den Automounter
auto.home	auto.home	Konfiguration für den Automounter

Tabelle 45.1: Übersicht der Map Namen in einer NIS Umgebung

45.7 Map Dateien

Eine Mapdatei muß vorher erzeugt werden. Dabei bedient sich das Programm, das die Maps erzeigt, den Quelldateien. Als Quelldatei kann jede beliebige Datei in frage kommen. Es gibt jedoch einige Maps die man besser vom Original ableiten sollte, wie z.B. die /etc/passwd. Die hosts muß nicht die /etc/hosts sein, das kann auch jede andere Datei sein.

Das Programm welches die Mapdateien erzeugt kann auch dahingehend Konfiguriert werden welche Maps der Server führen soll. Weiterhin ist auch nicht jeder Client in der Lage die Maps auszuwerten. Linux z.B. kümmert sich um die Map passwd.adjuct.byname überhaupt nicht.

45.8 Übersetzung in Maps

Die Übersetzung von Datenbankdateien in Mapdateien übernimmt ein Makefile. Make ist ein Programm welches nach bestimmten Bedingungen und Abhängigkeiten diverse Funktionen ausführt. Die eigentliche Konfigurationsdatei sollte man bis auf wenige Zeilen NICHT ändern. In den jeweiligen Kapiteln ist die Manipulation des Makefiles beschrieben.

45.9 Und grafisch siehts so aus

45.9.1 NIS Master Server

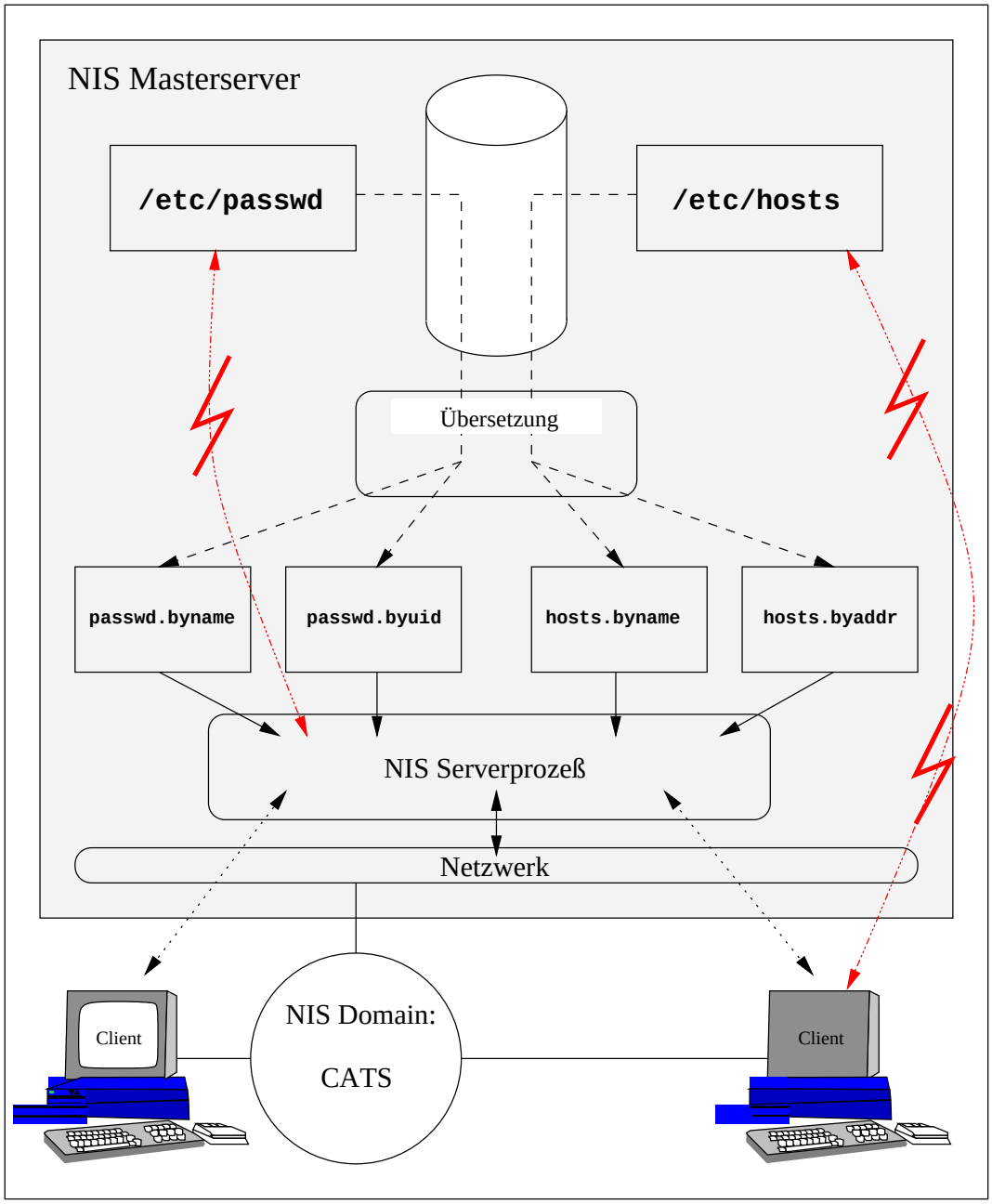

Abbildung 45.1: NIS Master Funktionsweise

45.9.2 NIS Master - NIS Slave Konzept

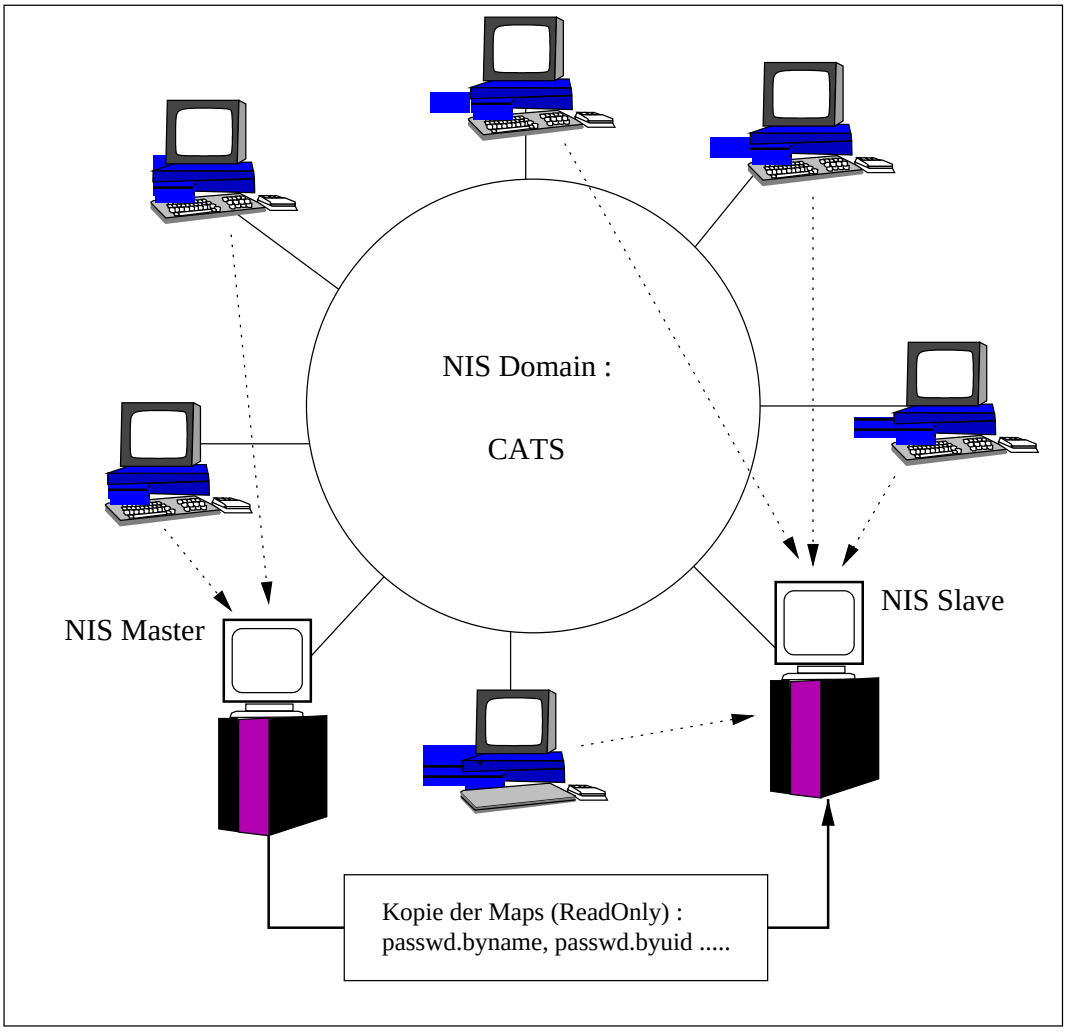

Abbildung 45.2: NIS Master - NIS Slave Funktionsweise

Kapitel 46

UNIX ALLGEMEIN KONFIGURATION

46.1 Konfiguration der Abfragereihenfolge

Fast jedes UNIX System verfügt über die /etc/nsswitch.conf die die Reihenfolge der Abfragen definiert.

/etc/nsswitch.conf

- Die /etc/nsswitch.conf ist eine gewöhnliche ASCII Textdatei
- Als Kommentarzeichen kann das Hash # verwendet werden, der nachfolgende Text wird bis zum Zeilenende ignoriert
- Änderungen in dieser Datei wirken sich sofort aus
- In jeder Zeile wird das Suchverhalten eines Dateischluessels beschrieben
- Die Informationen in einer Zeile werden durch Leerzeichen oder Tabulatoren getrennt.
- In der ersten Spalte steht der Dateischluessel um den es gehen soll. In der gesamten Datei darf dieser Schlüssel nur einmal auftauchen.
- Der Schlüssel ist von den Sucheinträgen durch ein Doppelpunkt zu trennen
- Ein Sucheintrag kann die folgenden Werte besitzen :

files	Sucht in den lokalen Datenbankdateien
nis	Fragt NIS Server
nisplus	Fragt NIS+ Server
dns	Fragt DNS Server, nur gültig im Schlüssel `hosts`
compat	Sucht in der Lokalendatenbank bis zum '+' Zeichen und fragt dann den NIS Server, nur erlaubt im Schlüssel `passwd` und `group`

- Weiterhin kann das Suchverhalten durch Ergebnisresultate gesteuert werden. Alle möglichen Ergebnisse einer Suche, sind in der folgenden Tabelle festgehalten

SUCCESS	Der gefragte Eintrag konnte besorgt werden. Default : return
UNAVAIL	Der Dienst der gefragt werden sollte erweist sich als Fehlerhaft. Möglich das die Verbindung zum Server gerade down ist. Default : continue
NOTFOUND	Der Server ist OK jedoch wurde der Eintrag nicht gefunden. Default : continue
TRYAGAIN	Der Server ist gerade beschäftigt. Die Anfrage sollte wiederholt werden. Default : continue

Auf diese Ereignisse kann man nun wie folgt Reagieren :

continue	Weiter mit dem nächsten Sucheintrag
return	Verlassen der Suche

- Das folgende Beispiel soll den Aufbau der Datei verdeutlichen :

Quelltext 46.1.1 Beispieleintragungen in der /etc/nsswitch.conf

```
passwd: files nis
group: nis
hosts: dns [NOTFOUND=return] files
```

Die Benutzer werden zuerst in der /etc/passwd gesucht. Werden die dort nicht gefunden wird NIS gefragt. Bei den Gruppen wird nur NIS gefragt, dort findet keine Suche in der /etc/group statt. WARNUNG: Wenn NIS nicht läuft oder der Server down ist kann sich keiner mehr Einloggen ! Man sollte dann ein Eintrag wählen der der `hosts:` gleichkommt. Dort wird zuerst DNS gefragt. Wurde der Eintrag nicht gefunden bis hierher und der DNS Server war erreichbar dann wird hier die Suche beendet. Es wird also verhindert das in der /etc/hosts gesucht wird. War der DNS nicht erreichbar wird noch die /etc/hosts durchsucht.

46.2 Programme und Konfigurationsdateien

Es folgt nun eine Liste von Programmen und Konfigurationsdateien die normalerweise in einer NIS Umgebung zur Verfügung stehen. In wie weit das einzelne UNIX diese Programme besitzt ist in den weiteren Kapiteln eventuell erklärt worden ....

Tabelle 46.1: Programme und Dateien in der NIS Umgebung

Programm / Datei	**Bedeutung**
ypinit	Konfigurationsprogramm für NIS Server und NIS Clients
yppush	Veranlasst den NIS Server die Maps neu einzulesen
ypxfr	Dient zur Übertragung von Maps. Wird meiset auf NIS Slaveservern eingesetzt
yppoll	Dient zur Abfrage welcher NIS Server welche Maps in welcher Domain verwaltet
ypset	Dient zur manuellen Anbindung an eine andere NIS Domain
ypwitch	Dient zur Anzeige an den aktuell angebundenen NIS Server
ypcat	Dient zur Ausgabe der Inhalte einer Map
ypmatch	Dient zur Anzeige von Inhalten einer Map die einen bestimmten Begriff beinhalten
yppasswd	Dient zum Ändern eines Passwortes über den NIS Server
domainname	Zeigt den Namen der aktuellen Domain an
ypserv	Implementiert den NIS Server Dienst
ypbind	Implementiert den NIS Client Dienst

▲

Programm / Datei	**Bedeutung**
ypxfrd	Serverdienst der nur für ypxfr benötigt wird
yppasswdd	Serverdienst der nur für yppasswd benötigt wird
ypupdated	Im Zusammenhang mit yppasswdd nötig
/var/yp/Makefile	Konfigurationsdatei für Mapdateien die der NIS Server zur Verfügung stellen soll
/var/yp/securenets	Nur den aufgelisteten Clients ist der Zugriff gewährt
/var/yp/nicknames	Aliasnamen für Mapdateien
/etc/netgroups	NIS Gruppen/Hosts Konfigurationsdatei
/etc/yp.conf	Konfigurationsdatei für NIS Clients

Kapitel 47

SPEZIELL LINUX ALLGEMEIN KONFIGURATION

47.1 Konfiguration der Maps

Als erstes muß man in das Verzeichnis /var/yp wechseln.

In der Datei /var/yp/Makefile steht beschrieben welche Dateien nun zu Maps gemacht werden sollen. Ausschlaggebend dafür ist eine Zeile die mit `'all:'` anfängt. Dort werden die Maps definiert.

Quelltext 47.1.1 Beispiel Eintrag in der /var/yp/Makefile

```
all:  passwd group rpc services protocols hosts networks shadow mail netid
```

Wobei `netid` immer angegeben werden muß. In dieser Datei befinden sich meist auch die Beschreibungen der anderen Map.

Nach den Anpassungen muß das Programm `make` aufgerufen werden. Dieses bearbeitet die Makefile. Vorher muß jedoch der Domainname und der NIS Server laufen.

Die Fehlermeldung

Quelltext 47.1.2 Zu ignorierende Fehlermeldung beim Übersetzen der Maps

```
failed to send 'clear' to local ypserv: RPC: Program not registered
```

kann ignoriert werden, sie deutet an das der NIS Server nicht läuft. Das Makefile informiert den laufenden NIS Server darüber das sich die Maps geändert haben, und wenn der NIS Server nicht läuft gibts ein Fehler !

Kapitel 48

SPEZIELL SUSE 6.X LINUX KONFIGURATION

48.1 Pakete für NIS Installieren

Zuerst müssen Die Pakete für NIS Installiert werden. Auf dem Server sollte man immer den NIS Server und den NIS Client installieren.

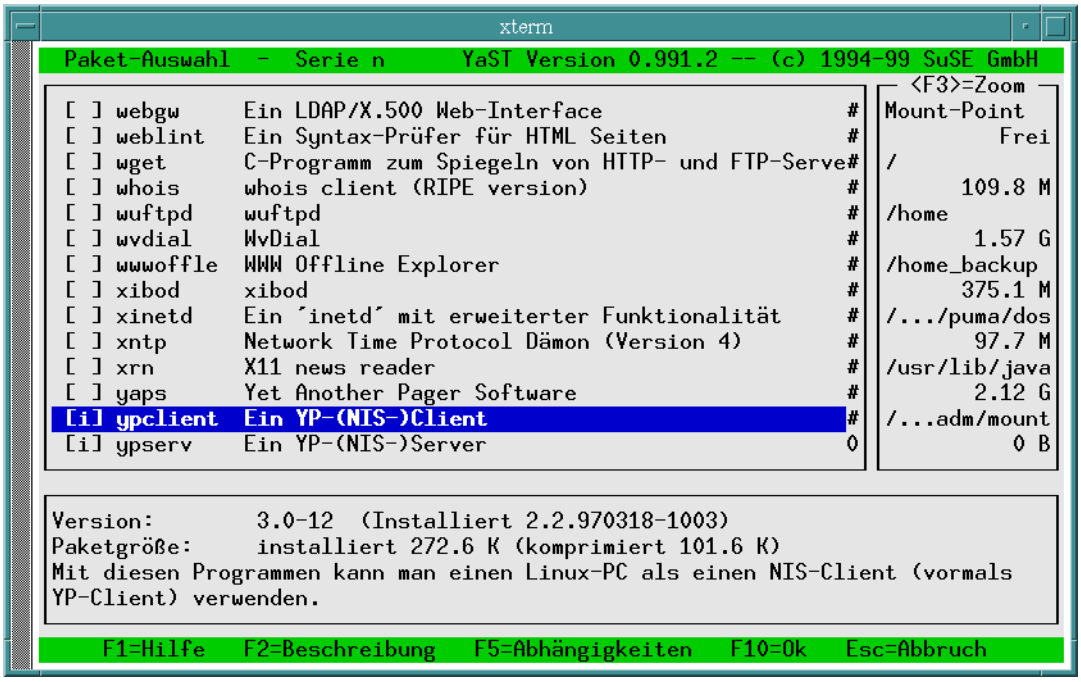

Abbildung 48.1: Paketauswahl vom YaST für NIS

Weiterhin muß noch unbedingt aus der Serie `d` - Developer das Paket `make` installiert werden !

48.2 Konfiguration des NIS Clients

Im YaST im Menu `Administration des Systems/Netzwerk konfigurieren/YP-Client konfigurieren` befindet sich die passende Maske

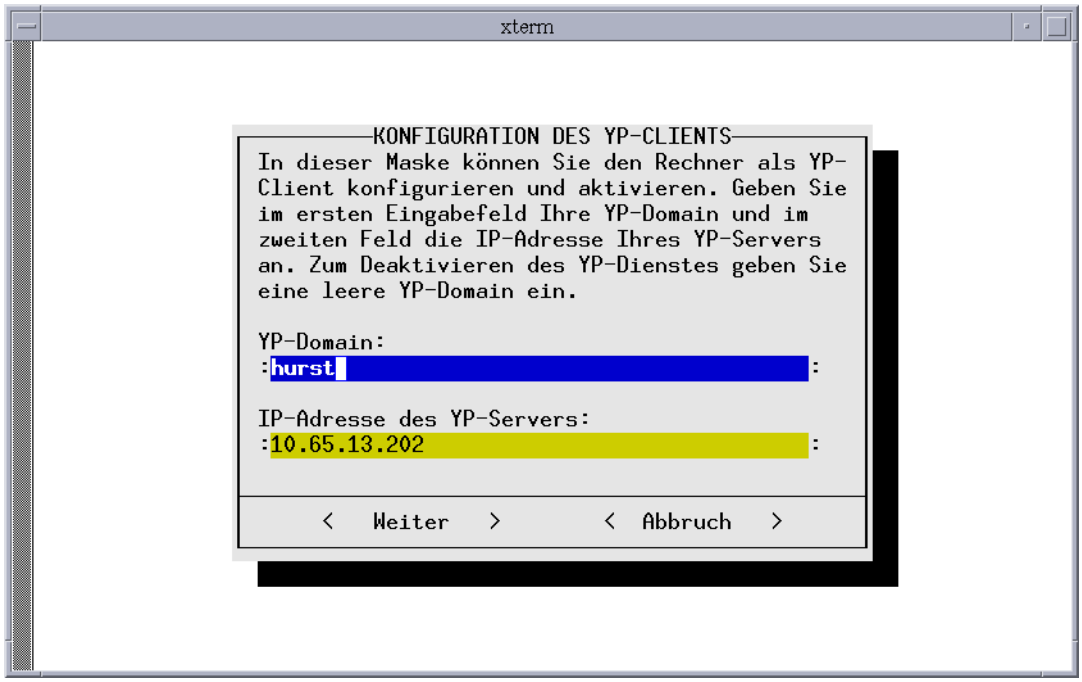

Abbildung 48.2: YaST Konfiguration eines NIS Clients

48.3 Konfiguration des NIS Servers

Als allererstes muß am NIS Server die NIS Client Konfiguration fertiggestellt sein. Als NIS Server IP kommt an die eigene ins Feld.

Danach ist in der /etc/rc.config zu Prüfen ob der Server überhaupt gestartet werden soll. Die folgenden Einträge sind dafür verantwortlich :

Quelltext 48.3.1 Einträge für NIS Server in der /etc/rc.config

```
START_YPSERV=yes
START_YPXFRD=no
START_YPPASSWDD=no
```

48.4 Starten und Stoppen des NIS Servers

Bevor der Server gestartet oder gestoppt werden soll muß unbedignt vorher geprüft werden, welcher NIS Domainname verwendet wird. Dazu ist die Angabe mittels `domainname` zu prüfen. Gegebenfalls kann der Domainnamen mit `domainname` neu gesetzt werden.

Bei der SuSE liegt das entspechende Startmodul in `/sbin/init.d/ypserv`. Das Startmodul kann mit `start` oder `stop` entsprechend Aufgerufen werden.

48.5 Konfiguration der Maps

siehe Linux Allgemein.

48.6 Starten und Stoppen des NIS Clients

Bevor der Client gestartet oder gestoppt werden soll muß unbedignt vorher geprüft werden, welcher NIS Domainname verwendet wird. Dazu ist die Angabe mittels `domainname` zu prüfen. Gegebenfalls kann der Domainnamen mit `domainname` neu gesetzt werden.

Bei der SuSE liegt das entspechende Startmodul in `/sbin/init.d/ypclient`. Das Startmodul kann mit `start` oder `stop` entsprechend Aufgerufen werden.

Testen kann man den NIS Client indem man einfach mittels `ypcat` eine Map runterläd.

Kapitel 49

SPEZIELL SOLARIS KONFIGURATION

Kapitel 50

PRAKTIKUM

50.1 First Try

Ziel : Jede Sitzreihe definiert ein NIS Server und konfiguriert die NIS Clients.
Jede Reihe arbeitet für sich allein :

1. Wählen Sie für Ihre Reihe ein NIS Domainnamen.
2. Wählen Sie einen Rechner für den NIS Server aus
3. Installieren Sie auf dem NIS Server das NIS Server Paket und auf den Clients das NIS Client Paket
4. Konfigurieren Sie nun alle Hosts als NIS Client, tragen Sie dazu in der Maske die notwendigen Werte ein
5. Prüfen Sie auf allen Hosts die /etc/nsswitch.conf darauf hin, daß in den Schlüsseln `passwd:` und `group:` zuerst der NIS danach die Localedatenbank abgefragt wird. Siehe 46.1 auf Seite 199
6. Stellen Sie in der /etc/rc.config sicher das der NIS Server auch gestartet werden kann.
7. Setzen Sie den Domainnamen manuell mittels `domainname`
8. Starten Sie den NIS Server
9. Konfigurieren Sie nun den NIS Server so das dieser die Maps `passwd`, `group` und `netid` exportiert.
10. Übersetzen Sie die Maps
11. Starten Sie auf den Hosts den NIS Client
12. Prüfen Sie mittels `ypcat passwd` ob der NIS Server die Map korrekt führt
13. Weiter gehts mit der nächsten Aufgabe ...

50.2 Der NIS User

1. Denken Sie sich ein Usernamen aus, der bisher auf keinem Host existiert. Ich werde hier einfach mal den Usernamen *otto* gebrauchen.
2. Versuchen Sie sich nun als Benutzer *otto* einzuloggen, egal welchen Host Sie nehmen, es funktioniert nicht.

3. Legen Sie auf dem NIS Server nun den Benutzer *otto* an. Vergeben Sie noch nicht das Passwort !
4. Prüfen Sie mittels `ypcat passwd` die Datenbank des NIS Servers. Der User taucht nicht auf. Warum ?!?
5. Genau. Sie müssen auf dem NIS Server die Maps neu übersetzen !
6. Prüfen Sie nach der Übersetzung nochmal die Maps mittels `ypcat`. Der User sollte nun auftauchen.
7. Versuchen Sie sich als Benutzer *otto* einzuloggen. Egal was Sie machen es geht nicht, weil das Passwort noch nicht hinhaut.
8. Vergeben Sie nun auf dem NIS Server ein Passwort für den Benutzer *otto*
9. Versuchen Sie sich nochmal einzuloggen. Jetzt klappt es nur auf dem NIS Server, jedoch nicht auf den anderen Hosts. Warum ?!?
10. Genau. Das Password wird mit der passwd-Map geführt. Da sich das Passwort geändert hat muß die Map neu Übersetzt werden !
11. Nach der Übersetzung sollte das Einloggen von allen Hosts dann auch möglich sein.
12. Die Fehlermeldung 'Homedirectory HOME=/ ...' kommt daher weil auf den anderen Hosts das Heimatverzeichnis nicht existiert. Exportieren Sie es doch einfach.
13. Exportieren Sie das Heimatverzeichnis des Benutzers *otto* auf dem NIS Server
14. Mounten Sie auf den Clients das Heimatverzeichnis des Benutzers *otto*
15. Nun sollte die Fehlermeldung auch nicht mehr erscheinen. Wenn Sie sich nun alle Gleichzeitig als *otto* anmelden und einer von Ihnen eine Datei im Heimatverzeichnis erstellt ist es bei den anderen sofort auch zu sehen. Und ab sofort ist es dem User auch vollkommen egal an welcher Maschine er sich einloggt.

50.3 The Final

Mit dieser Übung können Sie nur beginnen, wenn alle anderen bis hierher alle Übungen gemacht haben. Wenn Sie als erster fertig waren helfen Sie den anderen noch mit den anderen Übungen.

1. Löschen Sie den Benutzer auf allen NIS Server.
2. Stoppen Sie alle NIS Clients danach alle NIS Server
3. Ändern Sie die /etc/rc.config so das der NIS-Server nicht mehr gestartet wird
4. Jetzt lösen Sie sich von Ihrem Subnetz. Wir wollen ein NIS Server für die ganze Klasse installieren. Bitte denken Sie an das Zauberwort Teamarbeit bei der Bearbeitung dieser Übung !
5. Denken Sie sich dazu einen NIS Domainnamen aus.
6. Wählen Sie aus der gesammten Klasse ein Primären NIS Server.

7. Konfigurieren Sie auf allen Hosts den NIS Client. Setzen Sie den Domainnamen manuell via `domainname`
8. Installieren Sie auf dem NIS Server nun die NIS Server Pakete. Machen Sie den NIS Server startfähig und starten diesen. Danach starten Sie alle NIS Clients
9. Editieren Sie nun die Konfiguration des NIS Servers und geben die `passwd`, `group` und `netid` frei.
10. Übersetzen Sie nun die Maps
11. Erstellen Sie auf dem NIS Server eine Gruppe mit dem Namen Ihrer Klasse.
12. Übersetzen Sie die Maps und prüfen das Ergebnis auf allen Clients via `ypcat`
13. Erstellen Sie nun auf dem NIS Server für jeden Teilnehmer aus der Klasse einen eigenen Account. Vergeben Sie auf dem NIS Server auch ein Passwort. Die Benutzer sollten das Heimatverzeichnis unter /home haben und der o.g. Gruppe angehören.
14. Übersetzen Sie nun die Maps neu. Alle Benutzer sollten sich nun via Telnet auf den NIS Server einloggen und Ihr Passwort ändern.
15. Nachdem alle Passwörter geandert wurden übersetzt der Admin nocheinmal die Maps
16. Konfigurieren Sie nun den NIS Server als NFS Server und geben das Verzeichnis /home frei.
17. Alle NIS Clients mounten nun das /home vom Server nach /home. Tragen Sie diesen Mount in die /etc/fstab ein, so das das verzeichnis automatisch beim Booten gemountet wird.
18. Versuchen Sie noch ein Probelogin auf Ihrem Host. Starten Sie kurz X11 und konfigurieren irgendetwas um. Verlassen Sie danach das X11.
19. Gehen Sie zu einer vollkommen fremden Maschine und versuchen sich dort einzuloggen. Es sollte sich die gleiche Oberfläche auf tun.
20. Damit das ganze auch Niet- und Nagelfest wird Rebooten Sie Ihren NIS Client und testen das Verhalten nach dem Reboot.
21. Nachdem das alles geklappt hat rebooten Sie den NIS Server, um dessen Funktionalität zu prüfen.

Für den weiteren Unterricht ist der NIS/NFS Server wichtig. Es ist dafür sorge zutragen das dieser Rechner verfügbar ist, auch dann wenn der betreffende Admin nicht da ist.

TEIL XI

R-TOOLS

- Einführung

Kapitel 51

EINFÜHRUNG

Die R-Tools sind sehr mächtige Werkzeuge die ein normaler Benutzer in einem UNIX System gebrauchen kann. Der Admin *root* kann die Werkzeuge nur sehr bedingt einsetzen.

Mit den RTools ist es möglich sich entfernt einzuloggen, wie Telnet auch, jedoch können die RTools so konfiguriert werden, daß eine Passwordabfrage überflüssig wird. Das kann dann dazuführen daß man Prozesse innerhalb eines Shellscripts auf andere Rechner verteilen kann.

Die R-Shell erlaubt sogar das Pipen von Kommandos von anderen Rechnern, Telnet kann dieses nicht. Beispiel: Auf einem Host wird ein Backup gestartet. Während man tar benutzt und es auf dem Host A startet gibt tar die Ausgabe nicht in eine Datei sondern über die Pipe. Jetzt könnte man ein Komprimierungsprogramm wie gzip z.B. auf Host B starten. Die Daten von tar laufen dann auf den Host B zum GnuZippen. Weiterhin kann dann die Ausgabe von gzip über eine Pipe weiter zum Host mit dem Bandlaufwerk befördert werden. Die Möglichkeiten sind hier unbegrenzt.

Weiterhin können X11 Anwendungen auf anderen Hosts gestartet werden und dessen Ausgabe auf den eigenen Desktop gebracht werden. Somit kann ein zentraler XApplication Server irgendwo stehen.

Oder man will einfach nur die Rechenleistung des Nachbarrechners ausnutzen.

TABELLENVERZEICHNIS

ABBILDUNGSVERZEICHNIS

BILDSCHIRMAUSSCHNITTSVERZEICHNIS

QUELLTEXTVERZEICHNIS

ABKÜRZUNGSVERZEICHNIS

ARP . . . . . Address Resolution Protocol(7)
ARP . . . . . Address Resolution Protocol(9)
ARPA . . . . . Advanced Research Project Agency(1)
BIND . . . . . Berkeley Internet Name Domain(127)
CRC . . . . . Cyclical Redundancy Check(10)
CSMA/CD . . . . . Carrier Sense Multiply Access/Collision Detect(4)
Datagram . . . . . Daten die durch den Internet Layer laufen(23)
DNS . . . . . Domain Name System(117)
DoD . . . . . Departmanet of Defence(1)
FQDN . . . . . Full Qualified Domain Name(123)
FTP . . . . . File Transfer Protocol(115)
Gateway . . . . . Allgemeines Wort für einen Multiprotocol Router(75)
GGP . . . . . Gateway-Gateway Protocol(30)
Hops . . . . . Anzahl der dazwischen liegenden Router(75)
http . . . . . Hypertext Transferprotocol(168)
ICMP . . . . . Internet Control Message Protocol(33)
IGMP . . . . . Internet Group Management Protocol(45)
IHL . . . . . Internet Header Length(28)
IL . . . . . Internet Layer(4)
IP . . . . . Internet Protocol(4)
IPv4 . . . . . Internet Protocol Version 4(4)
IPv6 . . . . . Internet Protocol Version 6(4)
IPX . . . . . Internet Packet Exchange(69)
ISN . . . . . Initial Sequence Number(56)
ISN . . . . . Initialize Sequence Number(59)
ISO . . . . . International Standard Organisation(3)
MAC . . . . . Media Access Control(9)
MSS . . . . . Maximum Segment Size(57)
MTU . . . . . Maximum Transfer Unit(4)
NAL . . . . . Network Access Layer(4)
NFS . . . . . Network Filesystem(173)
NIC . . . . . Network Information Center(115)
NIC . . . . . Network Interface Card(9)
NIS . . . . . Network Information Services(193)
OSI . . . . . Open System Interface(3)
PPP . . . . . Point To Point Protocol(4)
PUP . . . . . PARC universal packet protocol(31)
RARP . . . . . Reverse ARP(10)
RLZ . . . . . Revers Lookup Zone(140)
RR . . . . . Resource Record(131)
SLIP . . . . . Serial Line Interface Protocol(4)
SRI . . . . . Standfort Research Institute(115)

TCP/IP . . . . . Transmission Control Protocol / Internet Protocol(1)
TCP . . . . . Transmission Control Protocol(4)
TCP . . . . . Transmission Control Protocol(55)
tftp . . . . . Trivial File Transfer Protocol(167)
TL . . . . . Transport Layer(4)
ToS . . . . . Type of Service(28)
TTL . . . . . Time To Live(28)
UDP . . . . . User Datagram Protocol(4)
UDP . . . . . User Datagram Protocol(53)
WAN . . . . . Wide Area Network(4)
YP . . . . . Yellow Pages(193)

REQUEST FOR COMMENTS

INDEX