

Netzwerkadministration für TCP/IP

- Das Internet
- Adressen und Netze
- Protokollschichten
- Routing
- Das Domain Name System
- Anwendungen
- Mailing
- Andere Services

Das Internet

- entstanden aus dem Arpanet
- eines der größten weltweiten Netze
(z.Zt. über 1 Million Hosts, ca. 300000
in Europa)
- Netz von Netzen, die die Internet
Protokoll-Suite benutzen
(==> kein homogenes Netz)
- Internet Protokolle:
 - offen, Protokollspezifikationen
sind frei erhältlich
 - unabhängig von der physikalischen
Netzwerkhardware
 - einheitliches Adressierungsschema
 - standardisierte Anwendungs-
protokolle
 - Protokolle sind beschrieben in
RFCs (Request for Comments)
frei erhältlich

Protokollarchitektur

OSI 7 Schichten Modell

- 7 - Anwendungsschicht
- 6 - Darstellungsschicht
- 5 - Sitzungsschicht
- 4 - Transportschicht
- 3 - Netzwerkschicht
- 2 - Sicherungsschicht
- 1 - Physikalische Schicht

Die Internetprotokolle entsprechen dem OSI 7 Schichten Modell nur angenähert.

Die TCP/IP Protokoll Architektur läßt sich in einem 4 Schichten Modell darstellen:

- 4 - Anwendungsschicht
- 3 - Host-zu-Host Transportschicht
- 2 - Internetschicht
- 1 - Netzzugriffsschicht

Daten werden über Encapsulation durch die einzelnen Schichten gereicht.

Netzzugriffschicht - Network Access Layer

- Schnittstelle zum physikalischen Netzwerk
- IP Pakete werden Pakete des Netzes verpackt
- Abbildung der IP-Adressen auf Hardware-Adressen und umgekehrt
Protokolle:
 - ARP - Address Resolution Protocol (RFC 826)
 - RARP - Reverse Address Resolution Protocol (RFC 903)

Internetschicht - Internet Layer

Internet Protocol (IP):

- beschrieben in RFC 791
- Basisschicht
- Datagrammerstellung
- definiert Internet-Adressen
- Bestimmung von Leitwegen, Routing, Verwaltung von Tabellen
- Fragmentation und Rekonstruktion von Datagrammen, Internet Protokoll kann bis zu 64 Kbyte Datagramme handhaben. Ethernet, Token-Ring lassen aber nur Pakete bis zu etwa 2 KByte zu.
- verbindungsloses Protokoll, jedes Datagramm enthält Absender- und Zieladresse
- enthält keine Fehlererkennung und keine Fehlerbehandlung
- Weitergabe von Datagrammen an die Transportschicht - Identifikation über Protokollnummern

Internet Control Message Protocol (ICMP):

- beschrieben in RFC 792
- zu Kontrollfunktionen und Fehlermeldungen
- Flußkontrolle (Source Quench)
- Redirecting von Routen (Redirect)
- Verbindungstests (Echo)

Transportschicht - Transport Layer

User Datagram Protocol (UDP):

- beschrieben in RFC 768
- verbindungslos
- keine Fehlererkennung und keine Fehlerbehandlung
- Anwendungen:
 - einfache Systemservices
 - wenn Anwendung selber Fehlerbehandlung und -korrektur übernimmt
- Ressourcenfreundlich

Transmission Control Protocol (TCP):

- beschrieben in RFC 793
- verbindungsorientiert
- gesichert
- sehr aufwendig
- viele Optionen und Möglichkeiten
- starke Beanspruchung der Ressourcen
- Vielzahl der Anwendungen basieren auf TCP

Anwendungsschicht - Application Layer

- Terminal Emulation (TELNET, RFC 854)
- File Transfer (FTP, RFC 959)
- Mail (SMTP, RFC 821)
- Domain Name System (DNS, RFC 1034)
- Routing (RIP, RFC 1058)
- Network File System (NFS, RFC 1094)
- ...

Internet Adressen

- 32 Bit

- unterteilt in Netzwerkteil und Hostteil

- Unterscheidung von Klassen

- Klasse A

- Bit 1 0

- Bit 2-8 Netzwerk

- Bit 9-32 Host

- ==> weniger als 128 Netze

- jedes Netz - mehrere Millionen Hosts

- Klasse B

- Bit 1 1

- Bit 2 0

- Bit 3-16 Netzwerk

- Bit 17-32 Host

- ==> ca. 16000 Netze

- jedes Netz - ca. 65000 Hosts

- Klasse C

- Bit 1 1

- Bit 2 1

- Bit 3 0

- Bit 4-24 Netzwerk

- Bit 5-32 Host

- ==> ca. 2 Millionen Netze

- jedes Netz - 254 Hosts

- Klasse D

 - Bit 1 1

 - Bit 2 1

 - Bit 3 1

 - Multicastadressen für spezielle Internetanwendungen

- Schreibweise der Internetadressen

 - als Tupel von 4 Bytes (dotted quad)
a.b.c.d

 - Netzwerkadresse - alle Hostbits 0

 - Broadcastadresse - alle oder einige Hostbits 1

- Verwaltung und Adressvergabe geschieht durch Network Information Centers (NIC)

- Universität und Fachhochschule haben die Netzadresse 131.173.0.0 (Klasse B)

- wegen des großen Internetwachstums

 - Vergabe von Klasse B Adressen nur noch in genau begründeten Fällen (z.B. mehr als 4096 Hosts)

Subnetze

- zur weiteren Unterteilung eines IP-Netzes
- Überbrücken von Hardwareunterschieden
- zur organisatorischen Aufspaltung von Netzen
- Zugriffssteuerung
- Lasttrennung
- Überbrücken von Hardwareunterschieden
- zur organisatorischen Aufspaltung von Netzen
- Zugriffssteuerung
- Lasttrennung

- Aufteilung der Adressbits in Subnetzanteil und Hostanteil wird über die Subnetzmaske geteuert:

IP-Adress A und Maske M

$A \& M \implies$ alle Subnetzbits

$A \& (\text{nicht})M \implies$ alle Hostbits

Schreibweise als dotted quad

z.B. 255.255.255.0 entspricht

11111111.11111111.11111111.00000000

$\implies 131.173.17.1 \& 255.255.255.0$

= 131.173.17.0 als Subnetz

$131.173.17.1 \& 0.0.0.255$

= 0.0.0.1 als Host im Subnetz

- die Subnetzmaske steuert die Anzahl der IP-Netze
- die Subnetzmaske muß für ein ganzes Netz eindeutig sein
Universität und Fachhochschule haben 255.255.255.0 als Subnetzmaske

Konfiguration des Interfaces

UNIX: mit dem Kommando `ifconfig`

```
ifconfig <interface> <ip-address> \  
        netmask <netmask> broadcast \  
        <broadcastaddress>
```

```
<interface> - le0, ie0, lo0 ...  
<ip-address> - z.B. 131.173.17.1  
<netmask> - z.B. 255.255.255.0  
<broadcastaddress> - z.B.  
131.173.17.255
```

Anzeigen des Interface-Status mit:

```
ifconfig <interface> oder  
netstat -ain
```

Broadcastadresse - alle Hostbits des Subnetzes auf 1 setzen
(alte BSD UNIX-Varianten setzen alle Hostbits 0)

Zusätzliche Schlüsselwörter:

```
up, down - Interface an- und ausschalten  
arp, -arp - ARP benutzen oder nicht  
metric <n> - explizites Setzen einer Metric  
wird für Routing benutzt
```

PC/TCP: mit dem Kommando `ifconfig`
oder in der Sektion
`[pctcp interface n]` im
Konfigurationsfile

Neben der Internetadresse, der Netzmaske
und der Broadcastadresse werden einige
Adapterspezifische Daten konfiguriert.

Mit `ifconfig <device-file> show` oder
mit `inet stats` läßt sich der Status der
bekannte Interfaces abfragen.

Auflösung der IP Adressen - Address Resolution Protocol (ARP)

- zur Abbildung von IP-Adressen auf Hardwareadressen (z.B. Ethernet)
- beschrieben in RFC 826
- Aufbau von Adresstabellen in jedem Internethost.
- UNIX - Manipulation des ARP-Caches mit dem Kommando `arp`.
- PC/TCP - Anzeigen des ARP-Caches mit dem Kommando `inet arp`.

IP Konfiguration von PCs und diskless Workstations - Reverse Address Resolution Protocol (RARP) Bootstrap Protocol (BOOTP)

- RARP beschrieben in RFC 903
- BOOTP beschrieben in RFC 951
- erlauben die "dynamische" IP-Konfiguration anhand der Hardware-Adresse
- PC/TCP - BOOTP Implementation `bootp`.

Ports und Sockets

- IP-Protokoll benutzt Protocol Numbers zur Identifikation von Transport-Protokollen
- notwendig für die Benutzung von IP durch viele Transportprotokolle
- Protocol Numbers:

0	IP
1	ICMP
2	IGMP (Internet Group Multicast Pr.)
3	GGP (Gateway Gateway Pr.)
6	TCP
17	UDP
- UNIX verzeichnet in `/etc/protocols`
- TCP-Protokoll benutzt Ports zur Identifikation der Anwendungen
- 16 Bit $0 \leq \text{Port} \leq 65535$
- Ports ≤ 256 festgelegt für Standardanwendungen

21	FTP
23	TELNET
25	SMTP

- UNIX: $256 < \text{Port} \leq 1024$ Unix-Anwendungen
z.B. rlogin, rshell, talk usw.
- Ports > 1024 werden dynamisch vergeben,
wichtig für das Multiplexing von Verbindungen
z.B.:
Klient möchte Telnet-Sitzung zu einem
Server aufbauen. Dafür erhält er dynamisch
einen Quellport 3044 und einen Zielport 23.
Würde auch der Quellport 23 sein, könnte
niemand von einem anderen Rechner eine
Telnet-Sitzung bei dem Klienten beginnen!
- UNIX: Ports sind verzeichnet in
`/etc/services`
- Die Kombination von Port und IP Adresse
wird als Socket bezeichnet. Ein Socket
identifiziert eindeutig einen Netzwerkprozess
im gesamten Internet.
- Protocol Numbers und Ports sind
standardisiert. Aktuell RFC 1340.

Internet Routing

- das Internet ist ein Netz von Netzen
- wenn ein Paket von einem Netz in ein anderes übertragen werden muß müssen Rechner (Router - Gateways) benutzt werden, die Verbindung in mindestens 2 Netze haben
- Internet-Router sind der Klebstoff, der das Internet zusammenhält
- Routing wird über Tabellen gesteuert

Tabelleneintrag enthält:

- Zielrechner oder Zielnetzadresse
 - Adresse des Routers
 - zusätzliche Angaben, z.B. zur Bewertung der Route
- Routing-Algorithmus für jedes IP-Paket
 - 1) Hole Zieladresse aus Paket
 - 2) Für jeden Eintrag in der Routingtabelle: Vergleiche Zieladresse mit Zielangabe des Eintrages. Bei Übereinstimmung speichern der Adresse des Routers, bzw. Überschreiben einer bestehenden Adresse, wenn die Route günstiger ist

3) Router gefunden ==> sende Paket zu diesem Router

4) Router nicht gefunden
falls eine Defaultroute vorhanden ist
==> senden des Paketes an den angegebenen Router

falls keine Defaultroute vorhanden ist
==> FEHLER, PAKET KANN NICHT WEITERGELEITET WERDEN

- Schritt 3 funktioniert nur, wenn der Router direkt erreichbar ist, d.h. sich in dem Netz befindet, mit dem auch der Rechner verbunden ist.

- JEDER RECHNER IM INTERNET HAT EINE ROUTINGTABELLE

- UNIX:

auslesen der Routing-Tabelle mit
`netstat -rn`

Spalten

`Destination` - Zieladresse

`Gateway` - Adresse des Gateways

`Flags` - Zusatzinformation

`Refcnt` - Anzahl aktiver Nutzer

`Use` - Anzahl der versendeten Pakete

`Interface` - Benutzte Schnittstelle

Minimale Routing-Tabelle:

```
Routing Tables
Destination Gateway      Flags  Refcnt  Use  Interface
127.0.0.1   127.0.0.1   UH     1        132  lo0
131.173.17.0 131.173.17.7 U       20      4924  le0
```

Beide Einträge werden durch `ifconfig` beim Booten erzeugt

Eintrag 1 - Route zu localhost

Hostroute - Flag H

Eintrag 2 - Route zum direkt verbundenen Netz 131.173.17.0 durch das Interface `le0` mit der IP-Adresse 131.173.17.7 (mit `ifconfig` konfiguriert)

Statisches Routing:

Manuelle Einträge in der Routing-Tabelle
mit dem Kommando

```
route [-n,-f] {add,delete}  
        {net,host}  
        dest gw metric
```

-n - Ausgabe alle Einträge in Adressform

-f - Löschen der gesamten Tabelle

add, delete - Aktion

host, net - Host- oder Networkeintrag

dest - Zielnetz oder -rechner

gw - IP-Adresse des Routers

metric - Bewertung der Route

0 - Ziel ist sehr nahe
(direkt verbunden)

15 - Ziel ist weit weg
(unendlich)

Beispiel:

```
route add net 131.173.128.0 131.173.17.254 1
```

```
netstat -rn
```

```
Routing Tables
Destination Gateway      Flags  Refcnt  Use
Interface
127.0.0.1   127.0.0.1   UH     1       132   lo0
131.173.17.0 131.173.17.7 U       20      4924  le0
131.173.128.0 131.173.17.254 UG     0        0     le0
```

Eintragen der Defaultroute

Defaultroute - Eintrag für alle unbekanntenen Zielen

```
route add net default 131.173.17.254 1
```

```
netstat -rn
```

```
Routing Tables
Destination Gateway      Flags  Refcnt  Use
Interface
127.0.0.1   127.0.0.1   UH     1       132   lo0
131.173.17.0 131.173.17.7 U       20      4924  le0
default     131.173.17.254 UG     0        0     le0
131.173.128.0 131.173.17.254 UG     0        0     le0
```

Dynamisches Routing:

ICMP-Redirect

Beispiel:

```
ping 131.173.160.6
.
.
..... 0% packet loss
.
```

Dieser Rechner ist eigentlich nicht erreichbar, da kein Eintrag für das Netz oder den Host in der Routingtabelle vorhanden ist, sondern nur die Defaultroute.

Hier:

Per ICMP erhält der Rechner vom Defaultrouter mitgeteilt, daß der Zielrechner über einen anderen Router erreichbar ist (vorausgesetzt dem Defaultrouter ist dieses Gateway bekannt).

```
netstat -rn
```

```
Routing Tables
Destination Gateway      Flags  Refcnt  Use
Interface
127.0.0.1   127.0.0.1   UH     1       132   lo0
131.173.160.1 131.173.17.252 UGHD   1       10    le0
131.173.17.0 131.173.17.7  U     20      4924  le0
default     131.173.17.254 UG     0        0     le0
131.173.128.0 131.173.17.254 UG     0        0     le0
```

Dynamisch erzeugte Routen werden durch die D-Flag gekennzeichnet

Routing-Protokolle

Werden benötigt:

- für komplizierte Topologien
- in Backbones, wo Defaultrouten nicht mehr möglich sind
- in Netzen mit vielen Subnetzen
- interior und exterior Protokolle
 - in lokalen Netzen nur interior
 - in Weitverkehrsnetzen exterior
- sehr häufig wird verwendet RIP als interior Protokoll (RFC 1058)
`router` RIP Implementation

Algorithmus:

- 1) Alle 90 Sekunden verteilen alle Router in einem Netz ihre komplette Tabelle per Broadcast.
- 2) Lokale Rechner und andere Router führen Abgleich ihrer Tabelle mit diesen erhaltenen durch (mit Hilfe der mitgesandten Metric)

Vorteil:

- nur wenige Router müssen die Topologie des Netzes kennen

Nachteil:

- hohe Netzbelastung
- unnötige Information wird übertragen

Universität und Fachhochschule

ES WIRD NUR STATISCHES ROUTING
EINGESETZT.

JEDER RECHNER BESITZT EINE
DEFAULTROUTE ZUM NÄCHSTEN
ÄUßEREN GATEWAY (ZUM INTERNET
HIN).

JEDER RECHNER MUß ICMP-REDIRECT
VERSTEHEN (mittlerweile kein Problem)

Router sollten zentral gemanagt werden
(geht nicht immer)

ZENTRALER ROUTER:

131.173.17.254 (Ethernet)

131.173.128.254 (Token-Ring)

Konfiguration:

UNIX:

- /etc/routed in /etc/rc.local
NICHT starten (überflüssig)
- mit /etc/route Defaultroute setzen

PC/TCP:

- entweder
ifconfig <device-file>
 gw <Defaultrouter>
- oder im Konfigurationsfile
in der Sektion [PCTCP INTERFACE n]
router = <Defaultrouter>

Testen

UNIX:

```
ping -vs <IP-Adresse>
```

```
traceroute <IP-Adresse>
```

PC/TCP:

```
ping <IP-Adresse>
```

Das Domain Name System

- beschrieben in RFC 1034, 1035
- ersetzt die Datei `/etc/hosts`
- Eigenschaften:

Namen sind konsistent im Internet

Verteilte Datenhaltung

Allgemeine Nutzungsmöglichkeit

Erweiterbarkeit

Einfache Implementierung

Hierarchische Struktur des Namensraumes

.

edu

com

de

mit

udel

ibm

Uni-Osnabrueck

lcs

rz

physik

export

io

mne

Notation

- Punkte zwischen Nodes, z.B.
titan.rz.Uni-Osnabrueck.DE.
- von links nach rechts näher zur Root
- Gesamtlänge \leq 255 Zeichen
- Nodename/Subdomain \leq 63 Zeichen
- case insensitive
- 7-Bit Zeichen
- Top-Level Domain ISO Country Code,
z.B. DE, FR, SE, UK,
- Nodename beginnt mit Alpha-Zeichen
- sonst alphanumerisch und "-"
- KEINE BEZIEHUNG ZU IP-ADRESSEN

Angabe der Namen:

- Absolut: endend mit ".",
z.B. jupiter.rz.Uni-Osnabrueck.DE.
- Relativ: endend ohne ".",
z.B. jupiter.rz oder jupiter

Viele Unterschiede in den Implementationen

Zonen

- zusammenhängender Teil des Adressraumes
- bevollmächtigt zur Verwaltung aller Namen (authorisiert)
- Einrichtung ist mit der nächst höheren Autorität (z.B. für .DE) abzustimmen
- Verwaltung der Zonen verpflichtet sich zur Verlässlichkeit

Resource Records

- beschreiben Objekte im Domain Name Space
- Format:

Domainname Typ Daten

z.B.

```
$ORIGIN rz.Uni-Osnabrueck.DE.  
unios      A      131.173.17.7  
thebe      A      131.173.18.2
```

Format der Daten - abhängig vom Typ

- Typen:

A - IP-Adresse

NS - autoritativer Nameserver

CNAME - Aliasname

SOA - Start einer Autoritätszone

WKS - Well Known Service

PTR - Zeiger auf einen Domainnamen

HINFO - Host Information

MX - Mail Exchanger

Beispiele:

```
$ORIGIN Uni-Osnabrueck.DE.  
      NS unios.rz.Uni-Osnabrueck.DE.  
      NS amalthea.rz.Uni-Osnabrueck.DE.
```

```
$ORIGIN rz.Uni-Osnabrueck.DE.  
cisco CNAME neptun
```

```
@ IN SOA unios.rz.Uni-Osnabrueck.DE.  
  gtimmer.unios.rz.Uni-Osnabrueck.DE. (  
    1993031101 ; Serial  
    10800      ; Refresh 3 hours  
    3600       ; Retry 1 hour  
    3600000    ; Expire 1000 hours  
    86400)     ; Minimum 24 hours
```

```
$ORIGIN rz.Uni-Osnabrueck.DE.  
tethys WKS 131.173.17.10 TCP telnet ftp
```

```
$ORIGIN 17.173.131.IN-ADDR.ARPA.  
10 PTR tethys.rz.Uni-Osnabrueck.DE.
```

```
$ORIGIN rz.Uni-Osnabrueck.DE.  
unios HINFO SUN-IPC SunOS-4.1.2
```

```
$ORIGIN rz.Uni-Osnabrueck.DE.  
thebe MX 10 unios.rz.Uni-Osnabrueck.DE.
```

Abfragemechanismus:

Beispiel:

Wie lautet die IP-Adresse des Rechners
wuarchive.wustl.edu.?

1. Anfrage beim lokalen Nameserver
kein Eintrag ==> Anfrage bei
einem Root-Nameserver (z.B.
terp.umd.edu)
2. Antwort von terp.umd.edu
kein Eintrag, aber Nameserver für
wustl.edu. ist wugate.wustl.edu
(128.252.120.1) ==> Anfrage bei
wugate.wustl.edu.
3. Antwort von wugate.wustl.edu
IP-Adresse von wuarchive.wustl.edu ist
128.252.134.4

Iterativer Vorgang - jeder Nameserver
verweist auf den nächsten

Rekursiver Vorgang - lokaler Nameserver
leitet die Frage an den nächsten weiter,
führt aber die Beantwortung beim Fragenden
selber durch.

Software:

Server:

- UNIX Bind
(Berkeley Internet Name Domain)
/etc/named, /usr/etc/in.named

Resolver (Client):

- UNIX Library-Calls via libresolv.a
oder libc.a
erwartet wird /etc/resolv.conf

```
# Domain name resolver file
#
domain rz.Uni-Osnabrueck.DE
#
nameserver 131.173.17.7
nameserver 131.173.132.3
```

Implementationsabhängig:

Auswertung von `/etc/hosts`
DNS und YP(NIS)

SunOS (>= 4.1) :

```
/var/yp/Makefile
```

```
#B=-b  
B=
```

```
==>
```

```
B=-b  
#B=
```

Danach NIS-Map neu bauen und verteilen.

Abfragereihenfolge: NIS - DNS

Der gleichzeitige Betrieb von NIS und DNS erzeugt manchmal Probleme.

IBM AIX 3.2:

```
/etc/resolv.conf existiert ==>  
DNS wird benutzt. Abfragereihenfolge  
DNS, NIS, /etc/hosts
```

PC/TCP:

```
ipconfig <device-file>  
  ds 131.173.17.7 131.173.132.3
```

```
ipconfig <device-file>  
  domain rz.Uni-Osnabrueck.DE
```

```
ipconfig <device-file>  
  hostname hermes
```

oder

in der Sektion [pctcp general]

```
domain=rz.Uni-Osnabrueck.DE  
host-name=hermes
```

und in [pctcp addresses]

```
domain-name-server=131.173.17.7  
domain-name-server=131.173.132.3
```

Tools

- UNIX:

`nslookup` - Interaktives Befragen des
DNS (mächtig)

- PC/TCP:

`host` - Befragen des DNS

Universität und Fachhochschule:

Primary Nameserver: 131.173.17.7

Secondary Nameserver: 131.173.132.3

Mailing - Sendmail

Mail - Vielzahl von Transportprotokollen

- UUCP
- Bitnet
- SMTP
- X.400
- Novell MHS
- ...

TCP/IP - SMTP

- beschrieben in RFC 821 und RFC 822
- lesbares Protokoll auf Port 25
- beinhaltet kein User-Interface
wie `/bin/mail` `/usr/bin/mail` `elm`
`mush` `mh` ...

Beispiel für telnet auf port 25

```
220 amalthea.rz.Uni-Osnabrueck.DE Sendmail 5.61-AIX-1.2/2.01
ready at Tue, 23
HELO hermes.rz.Uni-Osnabrueck.DE
250 amalthea.rz.Uni-Osnabrueck.D Hello hermes.rz.Uni-
Osnabrueck.DE, pleased to
MAIL FROM: <gtimmer@rz.Uni-Osnabrueck.DE>
250 <gtimmer@rz.Uni-Osnabrueck.DE>... Sender ok
RCPT TO: <gtimmer@dosunil.rz.Uni-Osnabrueck.DE>
250 <gtimmer@dosunil.rz.Uni-Osnabrueck.DE>... Recipient ok
DATA
354 Enter mail, end with "." on a line by itself
Hallo, wie gehts.
.
250 Ok
QUIT
221 amalthea.rz.Uni-Osnabrueck.D closing connection
```

Generelles Problem:

Transportprogramm für eine Anzahl
von Transportprotokollen und Mailformaten

UNIX:

- sendmail
- smail
- zmailer
- MMDF
- PP

Sendmail - am weitesten verbreitet

sender 1 sender 2 sender 3

sendmail

mailer 1 mailer 2 mailer 3

Funktionsweise:

- beim Aufruf (direkt oder durch Programm)
 - Optionen verarbeiten
 - Empfängeradressen sammeln
 - Aliasnamen auflösen
 - Syntaxchecks

- Message speichern
 - Header im Memory
 - Body in temporärem File

- Message ausliefern
 - Welcher Mailer soll benutzt werden?
 - Adressen reformatieren
 - ...
 - Mailer-Programm aufrufen
 - alles oK ==> fertig
 - Fehler ==>
 - Queue Files aufbauen für
 - Body, Header und Control in
 - `/var/spool/mqueue`

- Fehlerfall
 - Sendmail sendet die komplette
 - Nachricht an den Absender zurück.

Konfiguration

- wird beschrieben durch ein einziges Konfigurationsfile `sendmail.cf`
- `cf` steht für Katastrophen-File

Syntax und Semantik:

- Interpretation einer Zeile hängt ab vom Inhalt der 1. Spalte
- Die Zeichen `<` `>` `(` `)` `"` `\` haben eine besondere Bedeutung. Änderungen sind hoffnungslos
- `#` und leere Zeilen - Kommentar
- Blank und Tab - Fortsetzungszeilen
- D - Variablendefinition
Format: `DName Wert`, z.B. `DMmailhost`
späterer Zugriff über `$Name`, z.B. `$M`

Eingebaute Variablen:

- a - Erzeugungsdatum der Mail im RFC-822 Format
- b - Gegenwärtiges Datum im RFC-822 Format
- c - Hop-Count der Mail (wie häufig sie durch diesen Rechner gelaufen ist)
- d - Gegenwärtiges Datum im UNIX ctime Format
- e - Die SMTP Startmeldung
- f - Userid des Absenders
- g - Absenderadresse relativ zum Empfänger (in der Regel die volle Mailadresse)
- h - Hostname des Empfängerrechners
- i - Queue ID
- j - offizieller Domain Name dieses Rechners
- l - Format der UNIX From-Zeile
- n - Name des Dämons (z.B. Mailer-Daemon)

- o - Satz der Trennzeichen für Parsing
(. ! : @ ^ % / [])
- p - Process ID von Sendmail
- q - Defaultformat der Absenderadresse
- r - Protokoll mit dem Sendmail angesprochen wurde (z.B. SMTP)
- s - Name des Remotehosts, der Sendmail anspricht
- t - Gegenwärtige Zeit - numerisches Format
- u - Username des Empfängers
- v - Version von Sendmail
- w - Hostname dieses Rechners
- x - Voller Name des Absenders
(z.B. Helmut Meyer)
- z - Home-Directory des Empfängers

- C - Klassendefinition
Format: CName wert1 wert2 wert3 ...
z.B. CT de au ca us se fr
späterer Zugriff über \$=Name (z.B. \$=T),
Bedeutung: Ist String in der Klasse enthalten

- F - Klassendefinition aus File
Format: FName Filename Format
z.B. FT/etc/toplevel %s
File wird mit scanf gelesen

- O - Optionen
Format: OOption Wert

Eingebaute Optionen:

A - Pfad des Aliasfiles (z.B /etc/aliases)

B - Ersetzungszeichen für Blanks in Adressen (z.B. .)

d - Zustellungsmodus

i (interactive) - sofort

b(background) - asynchron

q(queue) - durch Warteschlange

e - Fehlermeldungen

p(print) - Ausgabe am Terminal

q(quiet) - Nur Return Codes

m(mail) - Fehlermeldungen per Mail

w(write) - Fehlermeldungen an Terminal
oder per Mail

e(error) - Fehlermeldungen via Mail,
Exitcodes an Terminal

F - Filemode für temporäre Files (z.B. 0600)

H - Pfad für SMTP Help File

g - GID für die Mailer

L - Log-Level (für syslogd)

Q - Pfad des Queue-Directories

T - Queue Timeout, danach Mail nicht
zustellbar

u - UID für die Mailer

y, z, Z - Parameter für Prioritäten

$$\text{Prio} = n [\text{Bytes}] - (\text{Vorrangklasse} * z) \\ + (\text{Anz. Empfänger} * y)$$

bei jedem Queue-Durchlauf

$$\text{Prio} = \text{Prio} + Z$$

(Typisch: $0z1800$, $0y1000$, $0Z500$)

q, x, X - Parameter für Lastverhalten

$$\text{Wenn } \text{Prio} > q / (\text{Loadavg} - x + 1)$$

wird Mail nur gespoolt und nicht
ausgeliefert. Loadavg ist ein Maß
für die Belastung des Rechners

$$\text{Wenn } \text{Loadavg} > X$$

nimmt Sendmail keine Mail mehr entgegen

(Typisch: $0q10000$, $0x15$, $0X30$)

- P - Vorrangklassen

Format: PName=Wert, z.B. Pnormal=10

Jeder Benutzer kann eine dieser Klassen benutzen.

- T - Trusted User

Format: TUserid

Dürfen Mail für andere ausliefern, d.h.
Sender und From Adressen überschreiben

- H - Header Lines

Format: H?Mailer-Flag?Name:Format

z.B.

H?P?Return-Path: <\$g>

Wenn ein Mailer das P-Flag gesetzt hat,
wird eine Return-Path-Zeile in den Header
im angegeben Format eingefügt. Der
?..? Teil kann fehlen ==> die folgende
Zeile wird immer im Header eingefügt.

- M - Mailerdefinition
Format MName, Feld=Wert, Feld=Wert ...

Möglicher Felder:

P - Pfad, z.B. P=/bin/mail oder P=[IPC]

F - Flag, z.B. F=lsDFMe

S - Regelsatz für Absenderadresse,
z.B. S=10

R - Regelsatz für Empfängeradresse,
z.B. R=20

A - Argumentenstring für Mailer,
z.B. A=sh -c \$u

E - End-of-Line, z.B. E=\r\n

M - Maximallänge einer Message,
z.B. M=100000

Mailer-Flags:

D - Mailer benötigt Date: Zeile

C - Mailer fügt @domain zu Adressen hinzu

F - Mailer benötigt From: Zeile

M - Mailer benötigt Message-Id: Zeile

L - Zeilenlänge auf RFC-821 Wert beschränken

P - Mailer benötigt Return-Path: Zeile

R - Mail From: Zeile enthält Return-Path

X - Vor einzelnen . extra . einfügen

m - Mail an mehrere Empfänger gleichzeitig

s - "-Zeichen aus Adressen löschen

u - Groß- und Kleinschreibung von Usernamen beachten

h - Groß- und Kleinschreibung von Hostnamen beachten

n - keine UNIX From: Zeile in die Mail einfügen

- R - Rewriting Rules
Format: RPattern<Tab>Transformation

dienen zum Umschreiben von Adressen

Regel - Wenn die Adresse mit dem Pattern übereinstimmt, transformiere sie entsprechend der Transformation. Überprüfe die Regel solange, wie die Adresse mit dem Pattern übereinstimmt. Bei Nichtübereinstimmung nehme die nächste Regel.

Pattern-Matching basiert auf Token. Eine Adresse setzt sich aus einer Anzahl von Token zusammen. Trenner sind die in Do definierten Operatoren . ! @ usw.

Ein Pattern enthält Variablen, Klassen, Literale und spezielle Metasymbole.

Metasymbole können sein:

$\* - 0 oder mehr Token

$\$^+$ - 1 oder mehr Token

$\$$ - genau ein Token

$\$=x$ - ein Token aus der Klasse x

$\$\sim x$ - ein Token NICHT aus der Klasse x

$\$x$ - Token entspricht Variablen x

Beispiele für Pattern:

$\$+@\$+$

$\$+.\$+$

$\$+.\$=T$

$\$+@dosuni1.\$+$

Beim Matching werden Token einer Adresse in Variablen $\$1$, $\$2$, $\$3$... abgelegt.

Bei Mehrdeutigkeiten wird bis zur ersten Übereinstimmung abgespeichert.

Beispiel:

Adresse

harpo@rz.Uni-Osnabrueck.DE

```
$*      $1=harpo@rz.Uni-Osnabrueck.DE
$+      $1=harpo@rz.Uni-Osnabrueck.DE
$-      KEIN MATCH
$+@$+   $1=harpo,$2=rz.Uni-Osnabrueck.DE
$+.$+   $1=harpo@rz,$2=Uni-Osnabrueck.DE
$+.$=T  $1=harpo@rz.Uni-Osnabrueck,$2=DE
        (wenn DE in T)
$+@rz.$+ $1=harpo,$2=Uni-Osnabrueck.DE
```

Transformation

Enthält Literale, Variablen und Metasymbole

Metasymbole

$\$n$ - Wert der Variablen n

$\$[Name\$]$ - Kanonisierung von Name
(Nameserver)

$\$>n$ - Springe in Regelsatz n

$\$@$ - Beende Regelsatz

$\$:$ - Beende Regel

$\#\$$ - Setze Mailer, wird nur in Regelsatz 0
benutzt

Beispiele:

```
gtimmer@dosunil.bitnet
```

```
R$+@dosunil.bitnet  
    $1@dosuni.rz.Uni-Osnabrueck.DE
```

```
==> gtimmer@dosunil.rz.Uni-Osnabrueck.DE
```

```
R$+@$-.bitnet    $>10$1@$2.BITNET
```

```
==> gehe zu Regelsatz 10 mit  
    gtimmer@dosunil.BITNET als Input
```

```
R$+@$-.bitnet    $:$1@$2
```

```
==> beende Regel mit gtimmer@dosunil
```

```
R$-@R+          $:$1@$[$2$] # use nameserver
```

```
==> beende Regel mit gtimmer@dosunil.bitnet  
    denn .bitnet hat keine Einträge im Nameserver
```

```
R$+@$-.bitnet  
    $#tcp$@unios.rz.Uni-Osnabrueck.DE:$1@$2.bitnet
```

```
==> beende Regelsatz 0 mit (Mailer,Host,User) Tupel
```

```
R@              $#error$:Invalid Address
```

```
==> Fehlermeldung
```

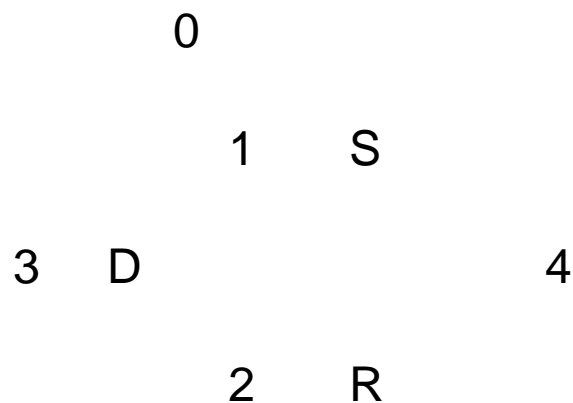
- S - Regelsätze

Regelsätze fassen eine Anzahl von Rewriting Rules zusammen

Format Sn ($0 \leq n \leq 29$)

Sn - Start von Regelsatz n, Ende ist vor dem Start des nächsten.

Globale Arbeitsweise von Sendmail wird durch eine Anzahl fester Regelsätze festgelegt.



3 - dient zur Transformation aller Adressen

D - Spezielle Teil, der für Absenderadressen von Mailer mit C-Flag durchlaufen wird Anhängen eines Domain-Parts für lokale Absenderadressen

0 - dient zur Bestimmung des (mailer,host,user) Tupels aus der Empfängeradresse

1 - dient zur Transformation der Absenderadresse

2 - dient zur Transformation der Empfängeradresse

S - Regelsatz aus dem S Mailer-Feld

R - Regelsatz aus dem R Mailer-Feld

4 - Transformation für alle Adresse

Die Konstruktion eines eigenen
`sendmail.cf` Files ist aussichtslos.

Einzigste Chance - Modifikation eines
bestehenden Files.

```

#
# Sendmail
# Copyright (c) 1983 Eric P. Allman
# Berkeley, California
#
# Copyright (c) 1983 Regents of the University of California.
# All rights reserved. The Berkeley software License Agreement
# specifies the terms and conditions for redistribution.
#
#
#####
#####
#####
#####          SENDMAIL CONFIGURATION FILE
#####
#####
#####
#####          Sendmail.cf fuer die Domaene
#####          Uni-Osnabrueck.DE
#####
#####          Meine deutschen Kollegen moegen mir
#####          verzeihen: All comments I made are given
#####          in English and are preceded by ---
#####
##### --- Remarks:
##### --- 1. Our domain is TCP/IP oriented
##### --- so I shot out all specific UUCP parts
##### --- 2. We want to use a main relay for
##### --- nonlocal mail
##### --- 3. We want to use a nameserver
##### --- 4. We want to make this file as easy as possible
##### --- 5. We want to make it as general as possible
##### --- 6. We want to make it maintainable
##### --- 7. BUT WE CANT GET ALL OF THE ABOVE TOGETHER
#####
#####
#####
#####
# --- host: all SUNs with SunOs 4.0 and greater 12.12.90

#####
### Setup Information ###
#####

# --- You have to setup some of this information
# --- I have marked them with a *** comment

#####
# General Macros #
#####

# local domain name
# ***
DDrz.Uni-Osnabrueck.DE

# relay host
DRunios.rz.Uni-Osnabrueck.DE

# my official hostname
Dj$w

#####
# Classes #
#####

# --- domainnames containing the following should go through our
# --- relay.

```

```
# --- DO NOT INSERT FULLY QUALIFIED DOMAIN NAMES.
# --- last update: 09.12.90 G.Timmer
# --- First all international Internet domains.
CTar at au be br ca ch cl cr de dk dd es fi fr gr hk ie il in is it
CTjp kr kw mx my ni nl no nz ph pr pt se sg su th tr tw uk us uy yu
# --- Next all american Internet domains.
CTarpa com edu gov mil net org
# --- Last but not least the pseudodomains.
CTbitnet cern dec hepnat int mfenet nato oz span uninett uucp wustl
```

```
#####
# Version Number #
#####
```

DZ2.01

```
#####
# Special macros #
#####
```

```
# my name
DnMAILER-DAEMON
# UNIX header format
DlFrom $g $d
# delimiter (operator) characters
Do.:%@!^=/[ ]
# format of a total name
Dq$g$x ($x)$
# SMTP login message
De$j Sendmail $v/$Z ready at $b
```

```
#####
# Options #
#####
```

```
# location of alias file
# ***
OA/etc/aliases
# wait up to ten minutes for alias file rebuild
# *** some sendmail didnt have this option
#Oa10
# substitution for space (blank) characters
OB.
# (don't) connect to "expensive" mailers
#Oc
# default delivery mode (deliver in background)
Odbackground
# temporary file mode
# *** some sendmails didnt have this option
OF0600
# default GID
Ogl
# location of help file
# ***
OH/usr/lib/sendmail.hf
# log level
OL9
# default network name
ONARPA
# default messages to old style
Oo
# queue directory
# *** usr or var
OQ/var/spool/mqueue
# read timeout -- violates protocols
Or2h
# status file
# ***
OS/usr/lib/sendmail.st
```

```
# queue up everything before starting transmission
Os
# default timeout interval
OTlh
# time zone names (V6 only)
# --- not conform to RFC822, but I hope it will work
# *** some sendmails didnt have this option
OtMEZ,MESZ
# default UID
Oul
# wizard's password
# *** some sendmails didnt have this option
OW*
# load average at which we just queue messages
Ox8
# load average at which we refuse connections
OX12
```

```
#####
# Message precedences #
#####
```

```
Pfirst-class=0
Pspecial-delivery=100
Pbulk=-60
Pjunk=-100
```

```
#####
# Trusted users #
#####
```

```
Troot
Tdaemon
Tuucp
```

```
#####
# Format of headers #
#####
```

```
H?P?Return-Path: <$g>
HReceived: $?sfrom $s $.by $j ($v/$Z)
            id $i; $b
H?D?Resent-Date: $a
H?D?Date: $a
H?F?Resent-From: $q
H?F?From: $q
H?x?Full-Name: $x
HSubject:
H?M?Resent-Message-Id: <$t.$i@$j>
H?M?Message-Id: <$t.$i@$j>
```

```
#####
### Rewriting Rules ###
#####
```

```
#####
# Sender Field Pre-rewriting #
#####
S1
```

```
#####
# Recipient Field Pre-rewriting #
#####
S2
```

```
#####
# Final Output Post-rewriting #
#####
```

S4

```
R@                @$                handle <> error   addr

# resolve numeric addresses to      name if      possible
R$*<@[$+]>$*      $:$1<@$[[2]]>$3      lookup numeric internet addr

# externalize local domain info
R$*<$+>$*        $1$2$3                defocus
R@$+:@$+:$+      @$1,@$2:$3          <route-addr> canonical

# delete duplicate local names
R$+%$=w@$=w      $1@$w                u%host@host => u@host
R$+%$=w@$=w.$D   $1@$w                u%host@host => u@host
```

```
#####
# Name Canonicalization #
#####
```

```
# --- Internal Format of names in rewriting rules is
# --- anything<@host.domain.domain...>anything
# --- We try to get every kind of name into this format,
# --- except for local names, which have no host part.
# --- The reason for the "<>" stuff is that the relevant
# --- hostname could be on the front of the name (for
# --- source routing), or on the back (normal form). We
# --- enclose the one that we want to route on in the <>'s
# --- to make it easy to find.
```

S3

```
# handle "from:<>" special case
R$*<>$*          @$@                turn into magic   token

# basic textual canonicalization -- note RFC733 heuristic here
# --- the next two rules are enclosed for compatability reasons
# --- so far I know
R$*<$*<$*<$+>$*>$*>$*   $4                3-level <> nesting
R$*<$*<$+>$*>$*        $3                2-level <> nesting

# --- START RFC822 PARSING
# --- look for route addresses
R$*<$+>$*        $2                basic RFC821/822 parsing
# --- the next      is mainly for our old BITNET fans
R$+ at $+        $1@$2            "at" -> "@"

# make sure <a,@b,@c:user@d> syntax is easy to parse -- undone later
# --- this is for comma list of domains in the route
R@$+,$+          @$1:$2            change all ", " to ":"

# localize and dispose of route-based addresses
# --- normally makes only <> pair around first route
R@$+:$+          @$>6<@$1>:$2      handle <route-addr>

# more miscellaneous cleanup
# --- simply return with list
R$+:$*;$+        @$1:$2;$3        list syntax
# --- also simply return
R$+:$*;$         @$1:$2;$         list syntax
# --- here we get what we really wants
R$+@$+           $:$1<@$2>        focus on domain
R$+<$+@$+>      $1$2<@$3>        move gaze right
R$+<@$+>         @$>6$1<@$2>      already canonical
# --- END RFC822 PARSING

# convert old-style addresses to a domain-based address
R$+^$+           $1!$2            convert ^ to !
R$-!$+           @$>6$2<@$1.UUCP> resolve uucp names
R$+.$-!$+        @$>6$3<@$1.$2>   domain uucps
```



```

R$+<@$->          $:$1<@[$2$]>          canonicalize into dom
R$+<@$->          $:$1<@2.$D>          if nameserver fails

# if not local, and not domain behind relay ask the nameserver
R$+<@$.~T>        $@$1<@[$2.$3$]>      user@host.domain
R$+<@[+]>         $@$1<@[2]>          already ok

```

S24

```

# put in <> kludge
R$*<$*>$*        $1$2$3            defocus
R$*               $:>3$1            now canonical form

# pass <route-addr>'s through
R<@$+>$*         $@<@[$1$]>$2        resolve <route-addr>

# map colons to dots everywhere.....
R$*:$*           $1.$2            map colons to dots

# output local host in user@host.domain syntax
R$-              $1<@$w>           user w/o host
R$+<@$=w>        $:$1<@$w>         this host
R$+<@$->         $:$1<@[$2$]>       canonicalize into dom
R$+<@$->         $:$1<@2.$D>       if nameserver fails

# if not local, and not domain behind relay ask the nameserver
R$+<@$.~T>        $@$1<@[$2.$3$]>    user@host.domain
R$+<@[+]>         $@$1<@[2]>        already ok

```

```

#####
### Rule Zero ###
#####

```

```

#####
#####
#####
#####          RULESET ZERO PREAMBLE
#####
##### The beginning of ruleset zero is constant through all
##### configurations.
#####
#####
#####
#####

```

S0

```

# first make canonical
R$*<$*>$*        $1$2$3            defocus
# --- the next rule came from Berkeley
# --- I dont understand why the call S3 again, because thats already
# --- done. For security I leaved it here.
R$+              $:>3$1            make canonical

# handle special cases
# --- for numeric IP addresses make a NS lookup
R$*<@[+]>$*      $:$1<@[$2]$>$3      numeric internet addr
# --- ok the nameserver didnt know try to call it
# --- this is a little bit insecure, isnt it?
R$*<@[+]>$*      $#tcp$@[2]$:$1@[2]$3  numeric internet spec

# --- put into the local S6 rule
R$+              $:>6$1
# --- make local delivery
R$-<@$w>         $#local$:$1
R@               $#error$:Invalid address handle <> form

```


Testen:

Test ob Sendmail sich meldet

```
Telnet localhost 25
```

```
·  
·  
SMTP - Kommandos  
·  
·
```

Test für den Versand von Mail

```
/usr/lib/sendmail -t -v
```

Test für Rewriting Rules

```
/usr/lib/sendmail -bt -Csendmail.cf
```

Sicherheit in TCP/IP Netzen

Internet - offenes Netz

Kann zu Sicherheitsproblemen führen

Informationen:

- CERT (anonymous ftp: cert.sei.cmu.edu /pub/cert_advisories)
- DDN (anonymous ftp: nic.ddn.mil /scc)
- RFC 1281 - A Guideline for the Secure Operation of the Internet
- David Curry - Improving the Security of your UNIX System

Problembereich - Passwörter

- ca. 80 % aller Sicherheitslöcher
- KEINE FREIEN LOGINS
(Guest-Login oder Logins mit bekannten
Passwörtern)
- KEINE SYSTEM-ACCOUNTS MIT
DEFAULT-PASSWÖRTERN
- MÖGLICHST KEIN TFTP
(auskommentieren in inetd.conf)

- Wahl eines Passwortes

kein Login-Name

keine Eigennamen

keine Information, die mit der Person in Verbindung stehen (Autokennzeichen, Initialen, Telefonnummern)

kein Tastaturkombinationen, d.h. rtzuiop

kein rückwärts buchstabierbaren Wörter

keine Passwörter nur aus Zahlen

keine Beispielpasswörter aus Büchern

Passwörter aus Gross- und Kleinbuchstaben bilden

Mindestens 6 Zeichen

Möglichst zufällige Verteilung von
einigen Buchstaben

z.B. 1 Buchstaben der Wörter eines
Satzes, z.B. We are red, we are white,
we are danish dynamite ==>
Warwewwadd

Passwörter müssen behaltbar sein,
(sonst werden sie aufgeschrieben)

Problem: User-Accounts, die selten
benutzt werden! (Am besten Löschen)
Benutzer sollten möglichst wenige
Accounts haben.

Löschen aller nicht benutzten Accounts.

Jeder Benutzer sollte die Last Login
Meldung beim Einloggen beachten.

Einige UNIX-Systeme: Passwort-Aging

Software:

`npasswd`, `passwd+`

Erzeugung von Passwörtern nach Regeln
ersetzen `passwd`

Zusätzliches Testen von Passwörtern mit
`crack` (z.B. nachts mit `cron`)

- Systemsoftware

Sind die Berkeley R-Kommandos
(`rlogin`, `rsh`, `rexec`) notwendig?

nicht ==> löschen der entsprechenden
Zeilen in `inetd.conf`

erzwingen einer Passwortabfrage ==>
löschen aller Files `.rhosts` aller
User und entsprechende Konfiguration
von `/etc/hosts.equiv` bzw. löschen

NFS - Benutzung der `-access` Option
aber `-root` nur in Ausnahmefällen in
`/etc/exports`

Secure Terminals in `/etc/ttytab` auf
`/dev/console` beschränken

Probleme mit `sendmail` < Version 5.65
(Remote Debugging)

Probleme mit `fingerd` < Ver. Nov. 1988

Probleme mit `ftpd` < Version 5.60

- Monitoring

COPS (Computer Oracle Password
and Security)

Reihe von Programmen und Shellscripts
die mögliche Sicherheitslöcher testen.

- Access Control:

Wrapper

Logging alle Versuche Internet
Services aus dem Netz zu starten.

Zugriffskontrolle, aller über `inetd`
gestarteten Internetservices.

Referenzen:

E. Nemeth, G.Snyder, S.Seebass:
UNIX System Administration Handbook
Prentice Hall (1989)

C. Hunt
TCP/IP Network Administration
Nutshell Handbook,
O'Reilly&Associates (1982)

D.Curry
UNIX System Security
Addison-Wesley (1982)

E.Krol
The Whole INTERNET
Nutshell Handbook,
O'Reilly&Associates (1992)

D.Curry
Improving the Security of your UNIX System
SRI International Report, April 1990

B.Kehoe
Zen and the Art of the Internet
Widener University, January 1992