wichtige Kommandos zum Netzwerkmanagement

- Kurzakte -

Diagnose-Tools für die einzelnen Hardwareschichtendes TCP/IP-Stacks

Schicht	Name	Protokolle	Diagnose-Tools
4	Anwendung	HTTP, FTP, SMTP, …	telnet, tcpdump, nslookup, dig, host, strace, ltrace
3	Verbindung	TCP, UDP, ICMP	netcat, netstat, nmap, lsof
2	Internet	IP,	ping, route, traceroute, ip
1	Hardware	Ethernet , ARP, PPP	arp, ifconfig

Management der Hardwareschicht

Netzwerkkarten sind die einzigen Geräte, die nicht unter /dev als Datei sichtbar sind. Sie werden von Treibern im Kernel realisiert und dort als Netzwerkgerät angesprochen.

`lspci –vvv` oder: `cat /proc/pci`
listet alle PCI-Geräte extrem ausführlich

`cat /proc/net/dev`
falls Netzkarten hier sichtbar, sind Treiberprobleme auszuschließen

`netstat –ai`
ähnliche Ausgabe

```
dmesg | grep –A 10 eth0
cat /var/log/messages | grep eth0
```
Listen der Fehlermeldungen beim Laden der Netzwerktreiber aus den Bootmeldungen
Anzeige der Geschwindigkeit (falls mii-tool nicht von Netzkarte unterstützt)

`ifconfig [-a]`
wird ein Gerät nicht gelistet, obwohl es unter /proc/net/dev sichtbar ist, muss es aktiviert werden
(Anzeige nicht aktiver devices mit ifconfig –a)
Deaktivieren eines Gerätes mit ifconfig eth1 down → I/Os und IRQ werden freigegeben

`ifup eth0 (ifdown,ifstatus)`
startet ein vorkonfiguriertes Interface, liest Daten aus /etc/network/interfaces

`ls /etc/sysconfig/network`
Anzeige aller durch Yast einzurichtenden Netzkarten beim Systemstart

`mii-tool`
Anzeigen des Mediums : 100 Mbit, der Übertragungsart: simplex, half duplex, duplex
des Linkstatus: link ok, no link Verhandlungsstatus: no autonegotiation, autonegotiation failed

`ethtool eth0`
Anzeige aller Netzwerkkarten-Einstellungen für eth0
`ethtool -S eth0 speed 10 duplex half autoneg off`
versetzt die Netzkarte eth0 in den Halb-Duplex-Modus bei 10er Geschwindigkeit ohne Aushandelungsprozedur.
`ethtool –r eth0` - startet Autonegotiation neu (cooler als das Netzkabel kurz zu ziehen)
`ethtool –p eth2` - lässt die LEDs von eth2 rhythmisch blinken (sinnvoll bei Routern)

Devices:

lo	Loopback-Device
sit0	zum Tunneln einer IPv6-Verbindung über IPv4
eth0	erste Ethernet-Netzwerkkarte
ppp0	Einwahlverbindung via Modem
ippp0	ISDN
pppoe	DSL-Verbindung
bond0	Bonding-Device

Herangehensweise bei Treiberproblemen mit Netzwerkkarten:
zuerst automatische Erkennung versuchen
dann manuelles Laden des Netzwerkkarten-Moduls ohne Parameter
erst jetzt Modul mitParametern laden (erst I/O-Adressen, dann IRQ u. a.)

Modulmanagement:

`insmod`	lädt Einzelmodule aus `/lib/modules/<Version>`
`rmmod`	entlädt Einzelmodule, die vom Kernel nicht mehr benötigt werden
`depmod`	erzeugt Datei /lib/modules/kernelversion/modules.dep mit den Modulabhängigkeiten (listen mit cat …/modules.dep | grep net | less)
`modprobe[-r]`	(ent-)lädt Module unter Berücksichtigung der Abhängigkeiten laut `/etc/modprobe.conf`, `/etc/modprobe.conf.local` und Verzeichnis `/etc/modprobe.d/` (ehemals `modules.conf`)
`lsmod`	listet die geladenen Module und von welchen anderen sie verwendet werden (Abhängigkeiten)
`modinfo`	zeigt die einkompilierten Modulinformationen an, inklusive der möglichen Parameter!!!

Möglichkeiten der Parameterübergabe (IRQ, DMA, Geschwindigkeit, usw.) :
1. am Bootprompt: "linux ether=5,0x300,eth0 ether=10,0x280,eth1" lädt 2 verschiedene Karten
   Syntax: ether=IRQ,BASE_ADDR, PARAM_1,PARAM_2,DEVICE_NAME (ohne Leerzeichen)
   IRQ IRQ-Kanal, der Wert 0 (Default) bedeutet AutoIRQ inklusive autoprobing
   BASE_ADDR Anfangsadresse des I/O-Bereichs, bei Wert 0 wird die kartenspezifische Adressliste wird durchsucht
   PARAM_1 spezielle Features einzelner Hersteller
   DEVICE_NAME eth0
2. Beim manuellen Laden des Moduls: "insmod e1000 io=0x300,0x280 irq=5,10" lädt 2 gleiche Karten
   Syntax: modprobe io=Wert1,Wer2,… irq=Wert1,Wert2,… (mit Leerzeichen)
3. in Datei /etc/modprobe.conf für jeden Treiber mindestens 2 Zeilen eintragen:
   install /eth0 /bin/true
   alias eth0 e1000 dem Treiber wird ein Gerätename zugewiesen
   options e1000 io=0x300 irq=10 Optionen für den Treiber
   alternativ in /etc/modprobe.conf.local (wird nicht von Yast überschrieben!!!) und im Verzeichnis /etc/modprobe.d

cat /proc/ksyms | grep e1000 zeigt Kernelunterstützung dieses Treibers an (Kernel-Symbol-Definitionen)

Hilfe über /usr/src/linux/Documentation/networking/net-modules.txt

Bsp.: 3 Netzwerkkarten gleichen Typs (Intel Pro100/10002 onboard, 1 als PCI-Karte)
einbauen, mit yast an alle Interfaces Adressen vergeben
aktives Netzwerkkabel an das Netzkarten-Interface einstecken
Reboot
IP des Interfaces mit ping checken

Neueste Treiber
Die neuesten Netzwerk-Karten-Treiber sind meistens von
http://www.scyld.com/network/
zu erhalten. Die Treiber werden anstelle der vorhanden Quelltexte in die passenden Verzeichnisse des Kernels kopiert und der Kernel neu übersetzt und installiert!

arp

Einsatz: Manipulation des arp-Caches

hier gelistste Hosts sind erreichbar
Aufdecken von ARP-Cache-Poisoning

Diagnosemöglicheiten:

Vergabe von 2 gleichen IP-Adressen

```
linux:~ # arp --help
Usage:
  arp [-vn]  [<HW>] [-i <if>] [-a] [<hostname>]             <-Display ARP cache
  arp [-v]          [-i <if>] -d  <hostname> [pub][nopub]    <-Delete ARP entry
  arp [-vnD] [<HW>] [-i <if>] -f  [<filename>]              <-Add entry from file
  arp [-v]   [<HW>] [-i <if>] -s  <hostname> <hwaddr> [temp][nopub] <-Add entry
  arp [-v]   [<HW>] [-i <if>] -s  <hostname> <hwaddr> [netmask <nm>] pub  <-''-
  arp [-v]   [<HW>] [-i <if>] -Ds <hostname> <if> [netmask <nm>] pub      <-''-

        -a                      display (all) hosts in alternative (BSD) style
        -s, --set               set a new ARP entry
        -d, --delete            delete a specified entry
        -v, --verbose           be verbose
        -n, --numeric           don't resolve names
        -i, --device            specify network interface (e.g. eth0)
        -D, --use-device        read <hwaddr> from given device
        -A, -p, --protocol      specify protocol family
        -f, --file              read new entries from file or from /etc/ethers

  <HW>=Use '-H <hw>' to specify hardware address type. Default: ether
  List of possible hardware types (which support ARP):
    tr (16/4 Mbps Token Ring)
    tr (16/4 Mbps Token Ring (New))
    arcnet (ARCnet) dlci (Frame Relay DLCI)
    fddi (Fiber Distributed Data Interface)
```

arp **listet den arp-Cache**

```
linux:~ # arp
Address                  HWtype  HWaddress           Flags Mask            Iface
172.16.0.244             ether   00:02:3F:DB:4C:23   C                     eth0
Flags: C - kompletter ARP-Eintrag
       M - manuell erstellter permanenter Eintrag (arp -s)
       P - publizierter Eintrag  (geliefert von einem Proxy Arp)
```

arp –a **Ausgabe im BSD-Format**

```
linux:~ # arp -a
? (172.16.0.244) at 00:02:3F:DB:4C:23 [ether] on eth0
server1 (172.16.0.2) at 00:4F:4E:0E:73:49 [ether] on eth0
```

cat /proc/net/arp **wie arp, Flags werden hexadezimal angezeigt**

```
linux:~ # cat /proc/net/arp
IP address       HW type     Flags       HW address            Mask     Device
172.16.0.244     0x1         0x2         00:02:3F:DB:4C:23     *        eth0
```

arp –n **bei Problemen mit der Namensauflösung**

arp –s <hostname> <MAC-Adresse> **manuelles Erzeugen eines ARP-Eintrags**

```
linux:~ # arp -s 172.16.0.88 00:02:3F:AA:BB:99
linux:~ # arp
Address                  HWtype  HWaddress           Flags Mask            Iface
172.16.0.88              ether   00:02:3F:AA:BB:99   CM                    eth0
```

arp –d <hostname> **Löschen eines ARP-Eintrags**

```
linux:~ # arp -v -d 172.16.0.88
arp: SIOCDARP(nopub)
linux:~ # arp
Address                  HWtype  HWaddress           Flags Mask            Iface
172.16.0.88                      (incomplete)                              eth0
```

arp –f <dateiname> **Einträge werden aus einer Datei eingelesen (Default: /etc/ethers)**

ifconfig

Einstaz: **Konfiguration der Netzwerkschnittstellen**
wird vom Kernel verwaltet und muss per Skript bei jedem systemstart realisiert werden

```
Usage:
  ifconfig [-a] [-i] [-v] [-s] <interface> [[<AF>] <address>]
  [add <address>[/<prefixlen>]]
  [del <address>[/<prefixlen>]]
  [[-]broadcast [<address>]]  [[-]pointopoint [<address>]]
  [netmask <address>]  [dstaddr <address>]  [tunnel <address>]
  [outfill <NN>] [keepalive <NN>]
  [hw <HW> <address>]  [metric <NN>]  [mtu <NN>]
  [[-]trailers]  [[-]arp]  [[-]allmulti]
  [multicast]  [[-]promisc]
  [mem_start <NN>]  [io_addr <NN>]  [irq <NN>]  [media <type>]
  [txqueuelen <NN>]
  [[-]dynamic]
  [up|down] ...

  <HW>=Hardware Type.
  <AF>=Address family. Default: inet
  List of possible address families:
    unix (UNIX Domain) inet (DARPA Internet) inet6 (IPv6)
    ax25 (AMPR AX.25) netrom (AMPR NET/ROM) ipx (Novell IPX)
    ddp (Appletalk DDP) x25 (CCITT X.25)
```

ifconfig **listet Einstellungen aller aktiven Schnittstellen (UP)**

```
linux:~ # ifconfig
eth0      Link encap:Ethernet  HWaddr 00:11:2F:BE:59:51
          inet addr:172.16.0.132  Bcast:172.16.0.255  Mask:255.255.255.0
          inet6 addr: fe80::211:2fff:febe:5951/64 Scope:Link
          UP BROADCAST NOTRAILERS RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:862056 errors:0 dropped:0 overruns:0 frame:0
          TX packets:460155 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:521208833 (497.0 Mb)  TX bytes:169001984 (161.1 Mb)
          Interrupt:5 Memory:d7efc000-0

lo        Link encap:Local Loopback
...
```

UP Interface ist aktiviert
RUNNING Interface funktioniert, sonst Treiberprobleme
Tipp zur Diagnose: RX packets sollte nicht 0 sein

ifconfig eth0 **listet Einstellungen von eth0**

ifconfig –a **listet alle Interfaces (UP and DOWN) – z.B. sit0**

ifconfig <interface> <ip-addresse> **Zuordnung einer IP zum Interface und Aktivierung**
alle anderen Parameter haben Default-Einstellungen

```
linux:~ # ifconfig eth0 172.16.1.2            # ergibt Netzmaske 255.255.255.0
```

ifconfig <interface> <ip-addresse> <netmask> up **Zuordnung einer IP mit Netzmaske und Aktivierung**

```
linux:~ # ifconfig eth0 172.16.1.2 netmask 255.255.255.0 up   # 24-bittige Netzmaske
```

ifconfig <interface> <ip-addresse> <netmask> <broadcast> up **vollständiger Befehl mit Broadcast**

```
linux:~ # ifconfig eth0 172.16.1.2 netmask 255.255.255.0 broadcast 172.16.1.255 up
```

ifconfig <device> hw ether <MAC> **Vergabe einer neuen MAC-Adresse**

```
linux:~ # ifconfig eth0 down
linux:~ # ifconfig eth0 hw ether 00:11:22:33:44:55
linux:~ # ifconfig eth0 up
```

ifconfig <device>:0 <zweit-ip-adresse> **Vergabe einer zweiten IP-Adresse**

ip

```
linux:~ # ip --help
Usage: ip [ OPTIONS ] OBJECT { COMMAND | help }
where  OBJECT := { link | addr | route | rule | neigh | tunnel |
                   maddr | mroute | monitor }
       OPTIONS := { -V[ersion] | -s[tatistics] | -r[esolve] |
                    -f[amily] { inet | inet6 | ipx | dnet | link } | -o[neline] }


linux:~ # ip addr [show]
1: lo: <LOOPBACK,UP> mtu 16436 qdisc noqueue
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 brd 127.255.255.255 scope host lo
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,NOTRAILERS,UP> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether 00:11:2f:be:59:51 brd ff:ff:ff:ff:ff:ff
    inet 172.16.0.132/24 brd 172.16.255.255 scope global eth0
    inet6 fe80::211:2fff:febe:5951/64 scope link
       valid_lft forever preferred_lft forever

linux:~ # ip addr add <ip-adresse/CIDR> dev eth0



linux:~ # ip route [show]
192.168.53.0/24 dev vmnet8  proto kernel  scope link  src 192.168.53.1
172.16.0.0/24 dev eth0  proto kernel  scope link  src 172.16.0.132
127.0.0.0/8 dev lo  scope link
default via 172.16.0.200 dev eth0

linux:~ # ip route add <ip-adresse> via <router>



linux:~ # ip link set dev eth0 up             aktiviert Interface eth0
```

route

Einsatz: **Erstellen der Routingtabelle des Kernels**
Achtung: Diese Einträge werden von oben nach unten abgearbeitet, bis der erste passt.
wird vom Kernel verwaltet und muss per Skript bei jedem Systemstart realisiert werden

```
linux:~ # route --help
Usage: route [-nNvee] [-FC] [<AF>]           List kernel routing tables
       route [-v] [-FC] {add|del|flush} ...  Modify routing table for AF.

       route {-h|--help} [<AF>]              Detailed usage syntax for specified AF.
       route {-V|--version}                  Display version/author and exit.

        -v, --verbose            be verbose
        -n, --numeric            don't resolve names
        -e, --extend             display other/more information
        -F, --fib                display Forwarding Information Base (default)
        -C, --cache              display routing cache instead of FIB

  <AF>=Use '-A <af>' or '--<af>'; default: inet
  List of possible address families (which support routing):
    inet (DARPA Internet) inet6 (IPv6) ax25 (AMPR AX.25)
    netrom (AMPR NET/ROM) ipx (Novell IPX) ddp (Appletalk DDP)
    x25 (CCITT X.25)
```

route **Anzeige die aktuellen Routingtabelle**

```
linux:~ # route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
172.16.0.0      *               255.255.255.0   U     0      0        0 eth0
loopback        *               255.0.0.0       U     0      0        0 lo
default         172.16.0.200    0.0.0.0         UG    0      0        0 eth0
```

Bemerkung: die Netzmaske 0.0.0.0 passt auf jedes Netz (ohne diesen Eintrag gibt es keine Internetverbindung)

Flags:			
	U	Up	Route ist inBetrieb
	H	Host	Ziel ist ein Host
	G	Gateway	Route benutzt einen anderen Rechner als Gateway
	D	Dynamic	dynamisch erzeugt durch Routing-Daemon
	M	Modified	geändert durch Routing-Daemon
	!	Rejected	durch den Router blockiertes Netz/Host (blocking route)

route add -net <Net-ID> Einrichten der Netzwerkkarte für das lokale Netz

- der Kernel sucht hierbei selbst nach der passenden Netzkarte

```
linux:~ # route add -net 192.168.1.0                            # Kurzfassung
linux:~ # route add -net 192.168.1.0 netmask 255.255.255.0 eth0  # vollständig
```

Achtung:
```
route add -net 172.16.1.0  # ohne –net macht Kernel ein -host daraus
route add 172.16.0.0       # Klasse B, alle Hostbits auf 0, → wird –net
```

Deaktivieren des Devices eth0

```
linux:~ # route del -net 192.168.1.0 netmask 255.255.255.0 eth0
linux:~ # ifconfig eth0 down
```

Hinzufügen einer weiteren Route zum Erreichen eines Netzes über einen Gateway

```
route add -net 172.16.0.0 netmask 255.255.255.0 gw romeo eth0        richtig
route add -net 172.16.0.0/24 gw romeo eth0                           richtig
route add -net 172.16.0.0/255.255.255.0 gw romeo eth0                falsch
```

Erstellen einer Default-Route

```
linux:~ # route add default gw 192.168.1.1                      # Kurzform
linux:~ # route add –net 0.0.0.0 netmask 0.0.0.0 gw 192.168.1.1  # Langform
```

Löschen der Default-Route

```
linux:~ # route del default gw 192.168.1.1 netmask 0.0.0.0
```

Erstellen einer "blocking route"

```
linux:~ # route add –net 10.0.0.0 netmask 255.0.0.0 reject
```

ping

Einsatz: **Senden von ICMP-Echo-Anforderungen an Netzwerkrechner**

```
linux:~ # ping --help
Usage: ping [-LRUbdfnqrvVaA] [-c count] [-i interval] [-w deadline]
        [-p pattern] [-s packetsize] [-t ttl] [-I interface or address]
        [-M mtu discovery hint] [-S sndbuf]
        [ -T timestamp option ] [ -Q tos ] [hop1 ...] destination
```

Diagnosemöglicheiten:

ping auf Loopback-Device	Kernel ist netzwerkfähig
ping auf eigene IP-Adresse	Netzwerkgerät ist aktiv und richtig konfiguriert
ping auf Standardgateway	Gateway ist online
ping auf Zielrechner	Zielrechner ist online und alle Gateways sind OK
eigenes Subnetz und Internet erreichbar, andere lokale Subnetze nicht erreichbar	falsche Netzmaske
langsame Antworten und Paketverluste (LAN)	Hardwareprobleme
langsame Antworten und Paketverluste (Internet)	normal

ping <Hostname|IP-Adresse> **Prüft Erreichbarkeit dieses Rechners**

```
linux:~ # ping 172.16.0.200
PING 172.16.0.200 (172.16.0.200) 56(84) bytes of data.
64 bytes from 172.16.0.200: icmp_seq=1 ttl=64 time=1.45 ms
64 bytes from 172.16.0.200: icmp_seq=2 ttl=64 time=1.61 ms

--- 172.16.0.200 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1000ms
rtt min/avg/max/mdev = 1.459/1.539/1.619/0.080 ms
```

ping –R Host **alle Hops werden mitnotiert**

ping –b Broadcastadresse **Erreichbarkeit aller Hosts auf einen Blick**
die Antwort auf Broadcast-Pings ist abstellbar über
`echo 1 > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts`

ping –c 3 –w 5 Host **nur drei Echo-Anforderungen senden und nach 5 sec abbrechen**
(nützlich für Skripte)

ping –s 65507 Host **geänderte Paketgröße (65507 statt 56 Bytes) erzeugt dummy traffic**

ping –a Host **ping wird hörbar über Lautsprecher**
(prima, wenn man gerade unter dem Tisch sitzt)

`ping -f Host` **Flood-Ping, nur von root absetzbar**

strace –e socket,connect,send,recvfrom ping www.zurquelle.de **Systemrufe zu ping**

```
linux:~ # strace -e socket,connect,send,recvfrom ping www.zurquelle.de
socket(PF_INET, SOCK_RAW, IPPROTO_ICMP) = 3
socket(PF_FILE, SOCK_STREAM, 0)         = 4
connect(4, {sa_family=AF_FILE, path="/var/run/nscd/socket"}, 110) = 0
socket(PF_FILE, SOCK_STREAM, 0)         = 4
connect(4, {sa_family=AF_FILE, path="/var/run/nscd/socket"}, 110) = 0
socket(PF_INET, SOCK_DGRAM, IPPROTO_IP) = 4
connect(4, {sa_family=AF_INET, sin_port=htons(53), sin_addr=inet_addr("217.237.149.225")}, 28) = 0
send(4, "\256\32\1\0\0\1\0\0\0\0\0\0\3www\tzurquelle\2de\0\0\1"..., 34, 0) = 34
recvfrom(4, "\256\32\201\200\0\1\0\1\0\0\0\0\3www\tzurquelle\2de\0\0"..., 1024, 0, {sa_family=AF_INET,
sin_port=htons(53), sin_addr=inet_addr("217.237.149.225")}, [16]) = 50
socket(PF_INET, SOCK_DGRAM, IPPROTO_IP) = 4
connect(4, {sa_family=AF_INET, sin_port=htons(1025), sin_addr=inet_addr("192.67.198.3")}, 16) = 0
PING www.zurquelle.de (192.67.198.3) 56(84) bytes of data.
socket(PF_INET, SOCK_DGRAM, IPPROTO_IP) = 4
connect(4, {sa_family=AF_INET, sin_port=htons(53), sin_addr=inet_addr("217.237.149.225")}, 28) = 0
send(4, "\256\33\1\0\0\1\0\0\0\0\0\0\0013\003198\00267\003192\7"..., 43, 0) = 43
recvfrom(4, "\256\33\201\200\0\1\0\1\0\0\0\0\0013\003198\00267\0031"..., 1024, 0, {sa_family=AF_INET,
sin_port=htons(53), sin_addr=inet_addr("217.237.149.225")}, [16]) = 73
64 bytes from web.webmailer.de (192.67.198.3): icmp_seq=1 ttl=246 time=28.9 ms
64 bytes from web.webmailer.de (192.67.198.3): icmp_seq=2 ttl=246 time=29.0 ms
PF_INET              Internetsocket
SOCK_DGRAM           UDP-Socket
connect              trotz connect wird keine Verbindung aufgebaut
```

traceroute

Einsatz: **Ausgabe der Router auf dem Weg zum Zielhost**

```
linux:~ # traceroute --help
traceroute: invalid option -- -
usage: traceroute [-nFV] [-f first_ttl] [-m max_hops] [-p port]
           [-S source_addr] [-I interface]
           [-t tos] [-w timeout] [-q nqueries] host [packetlen]
```

Funktionsweise:

Hinweg	je 3 UDP-Pakete mit anwachsender TTL (beginnend mit 1 statt 255)
Rückweg	Routerantworten: mit ICMP 11/0 (time exceeded), falls Router antworten dürfen Endgerät: mit ICMP 3/3 (port unreachable), weil hier kein Dienst lauscht
Problem	geht davon aus, dass jedes Paket von der Quelle bis zum Ziel immer den gleichen Weg nimmt, was im Internet nicht garantiert ist.

Standardeinstellungen:

- pro Router wird der Vorgang 3 mal wiederholt (ändern mit –q Anzahl)
- auf die Reaktion wird 3 Sekunden gewartet (ändern mit -w Sekunden)

```
traceroute –I eth1 www.zurquelle.de
```

netstat alternativ: socklist

Einsatz: **Anzeige von Netzwerkverbindungen, Routing-Tabellen, offenen Ports, Schnittstellen-Statistiken und Masquerade-Verbindungen**

```
linux:~ # netstat --help
usage: netstat [-veenNcCF] [<Af>] -r         netstat {-V|--version|-h|--help}
       netstat [-vnNcaeol] [<Socket> ...]
       netstat { [-veenNac] -i | [-cnNe] -M | -s }

        -r, --route              display routing table
        -i, --interfaces         display interface table
        -g, --groups             display multicast group memberships
        -s, --statistics         display networking statistics (like SNMP)
        -M, --masquerade         display masqueraded connections

        -v, --verbose            be verbose
        -n, --numeric            don't resolve names
        --numeric-hosts          don't resolve host names
        --numeric-ports          don't resolve port names
        --numeric-users          don't resolve user names
        -N, --symbolic           resolve hardware names
        -e, --extend             display other/more information
        -p, --programs           display PID/Program name for sockets
        -c, --continuous         continuous listing
        -l, --listening          display listening server sockets
        -a, --all, --listening   display all sockets (default: connected)
        -o, --timers             display timers
        -F, --fib                display Forwarding Information Base (default)
        -C, --cache              display routing cache instead of FIB
```

netstat **Listen aller offenen Verbindungen:**

1. Aktive (offene) Internetverbindungen

```
Aktive Internetverbindungen (ohne Server)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 192.168.32.178:ssh      192.168.32.181%3221:icp VERBUNDEN
```

Proto	das vom Socket benutzte Protokoll
Recv-Q	Anzahl der vom Client noch nicht abgeholten Bytes
Send-Q	Anzahl der von der Gegenstelle noch nicht bestätigten Bytes
Local Address	Lokale Adresse und Portnummer des Sockets ein * steht für alle verfügbaren Adressen des Servers
Foreign Address	Adresse und Portnummer der Gegenstelle
State	Zustand des Sockets (nicht für raw -Sockets und UDP): raw-Sockets und UDP-Sockets haben keine Zustände ESTABLISHED – Verbindung besteht CLOSE – Socket wird nicht verwendet LISTEN – Socket lauscht auf eingehende Verbindungen

```
ESTABLISHED The socket has an established connection.
SYN_SENT    The socket is actively attempting to establish a connection.
SYN_RECV    A connection request has been received from the network.
FIN_WAIT1   The socket is closed, and the connection is shutting down.
FIN_WAIT2   Connection is closed, and the socket is waiting for a shutdown from the remote end.
TIME_WAIT   The socket is waiting after close to handle packets still in the network.
CLOSED      The socket is not being used.
CLOSE_WAIT  The remote end has shut down, waiting for the socket to close.
LAST_ACK    The remote end has shut down, and the socket is closed. Waiting for acknowledgement.
LISTEN      The socket is listening for incoming connections.  Such sockets are not  included  in
            the output unless you specify the --listening (-l) or --all (-a) option.
CLOSING     Both sockets are shut down but we still don't have all our data sent.
UNKNOWN     The state of the socket is unknown.
```

2. Unix-Domain-Sockets:

```
Aktive Sockets in der UNIX Domäne (ohne Server)
Proto RefZÃ¤h Flaggen     Typ        Zustand       I-Node Pfad
unix  2      [ ]         DGRAM                    3216   /var/lib/ntp/dev/log
unix  12     [ ]         DGRAM                    3214   /dev/log
unix  2      [ ]         DGRAM                    41833
```

```
netstat -t   zeigt nur TCP-Verbindungen mit Status ESTABLISHED
netstat -ta  zeigt nur TCP-Verbindungen mit allen Stati
netstat -tl  zeigt nur TCP-Verbindungen mit Status LISTEN
```

Anzeige aller offenen (-l für listen) lokalen Ports für TCP (-t) und UDP (-u):

```
linux:~ # netstat -utl
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 *:netbios-ssn           *:*                     LISTEN
tcp        0      0 *:sunrpc                *:*                     LISTEN
tcp        0      0 *:telnet                *:*                     LISTEN
udp        0      0 *:xdmcp                 *:*
```

Achtung: verbundene Ports (ESTABLISHED) werden hierbei nicht angezeigt!

Anzeige aller offenen (-l für listen) lokalen Ports für TCP (-t) und UDP (-u) ohne Namensauflösung (-n), wobei das aufrufende Programm mit angezeigt wird:

```
linux:~ # netstat -utlnp
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
kded
tcp        0      0 0.0.0.0:139             0.0.0.0:*               LISTEN      4849/smbd
kded
tcp        0      0 0.0.0.0:111             0.0.0.0:*               LISTEN      4428/portmap
tcp        0      0 0.0.0.0:23              0.0.0.0:*               LISTEN      4944/xinetd
udp        0      0 :::177                  :::*                                4791/kdm
```

Unterschiede TCP zu UDP: Da bei UDP kein Verbindungsaufbau erfolgt, wird nicht zwischen Client und Server unterschieden. Jeder Socket kann immer senden und empfangen.

netstat –r Anzeige der Kernel-Routingtabelle (analog zu "route"):

```
linux:~ # netstat -r
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
172.16.0.0      *               255.255.255.0   U         0 0          0 eth0
loopback        *               255.0.0.0       U         0 0          0 lo
default         172.16.0.200    0.0.0.0         UG        0 0          0 eth0
```

netstat –r -C Anzeige des Kernel-Routingcaches

netstat –s [-t] Anzeige von Statistiken [auch protokollspezifisch]:

```
linux:~ # netstat -s
Ip:
    456092 total packets received
    0 forwarded
    0 incoming packets discarded
    453466 incoming packets delivered
    338041 requests sent out
Icmp:
    489 ICMP messages received ...
Tcp:
    95 active connections openings ...
Udp:
    33512 packets received ...
TcpExt:
    7 resets received for embryonic SYN_RECV sockets
    13 packets pruned from receive queue because of socket buffer overrun
    ArpFilter: 0
    75 TCP sockets finished time wait in fast timer ...
```

netstat –pantou alles Wichtige auf einmal:

```
linux:~ # netstat -pantou
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address Foreign Address  State   PID/Program name    Timer
tcp        0      0 0.0.0.0:901   0.0.0.0:*       LISTEN  4944/xinetd         off (0.00/0/0)
tcp        0      0 0.0.0.0:7     0.0.0.0:*       LISTEN  4944/xinetd         off (0.00/0/0)
tcp        0      0 0.0.0.0:5800  0.0.0.0:*       LISTEN  5244/kdeinit: kded  off (0.00/0/0)
tcp        0      0 0.0.0.0:139   0.0.0.0:*       LISTEN  4849/smbd           off (0.00/0/0)
```

netstat –i Anzeige der Interface-Statistik

netstat –tx Anzeige der aktiven Sockets für TCP und Unix-Domain-Sockets

lsof

Einsatz: **Anzeige von Informationen über offene Dateien**

```
linux:~ # lsof --help
usage: [-?abhlnNoOPRstUvV] [+|-c c] [+|-d s] [+D D] [+|-f]
 [-F [f]] [-g [s]] [-i [i]] [+|-L [l]] [+m [m]] [+|-M] [-o [o]]
[-p s] [+|-r [t]] [-S [t]] [-T [t]] [-u s] [+|-w] [-x [fl]] [--] [names]
Defaults in parentheses; comma-separate set (s) items; dash-separate ranges.
  -?|-h list help          -a AND selections (OR)     -b avoid kernel blocks
  -c c  cmd c, /c/[bix]    +c w  COMMAND width (9)
  +d s  dir s files        -d s  select by FD set     +D D  dir D tree *SLOW?*
                           -i select IPv[46] files    -l list UID numbers
  -n no host names         -N select NFS files        -o list file offset
  -O avoid overhead *RISK  -P no port names           -R list paRent PID
  -s list file size        -t terse listing           -T disable TCP/TPI info
  -U select Unix socket    -v list version info       -V verbose search
  +|-w  Warnings (+)       -- end option scan
  +f|-f  +filesystem or -file names
  -F [f] select fields; -F? for help
  +|-L [l] list (+) suppress (-) link counts < l (0 = all; default = 0)
                                      +m [m] use|create mount supplement
  +|-M   portMap registration (-)     -o o   o 0t offset digits (8)
  -p s   select by PID set            -S [t] t second stat timeout (15)
  -T qs TCP/TPI Q,St (s) info
  -g [s] select by process group ID set and print process group IDs
  -i i   select by IPv[46] address: [46][proto][@host|addr][:svc_list|port_list]
  +|-r [t] repeat every t seconds (15); + until no files, - forever
  -u s   exclude(^)|select login|UID set s
  -x [fl] cross over +d|+D File systems or symbolic Links
  names  select named files or files on named file systems
Anyone can list all files; /dev warnings disabled; kernel ID check disabled.
```

lsof –i **Anzeige *aller* über eine Netzwerkverbindung geöffneten Dateien (Port öffnende Prozesse)**

```
linux:~ # lsof -i
COMMAND     PID   USER   FD   TYPE DEVICE SIZE NODE NAME
dhcpcd     4104   root    6u  IPv4   7568       UDP *:bootpc
nmbd       4477   root    9u  IPv4   8630       UDP *:netbios-ns
sshd       4777   root    3u  IPv6   9151       TCP *:ssh (LISTEN)
...
sshd      15307   root    3u  IPv6 475796       TCP 172.16.0.132:ssh->172.16.0.244:neod1 (ESTABLISHED)
```

lsof –i [protocol][@hostname|hostaddr][:service|port] **nur von dieser Adresse geöffnete Dateien**

```
linux:~ # lsof -i @172.16.0.132
COMMAND   PID USER   FD   TYPE DEVICE SIZE NODE NAME
nmbd     4477 root   17u  IPv4 467997       UDP 172.16.0.132:netbios-ns
nmbd     4477 root   18u  IPv4 467998       UDP 172.16.0.132:netbios-dgm
sshd    15307 root    3u  IPv6 475796       TCP 172.16.0.132:ssh->172.16.0.244:neod1 (ESTABLISHED)
sshd    15466 root    3u  IPv6 482923       TCP 172.16.0.132:ssh->172.16.0.244:neod2 (ESTABLISHED)

linux:~ # lsof -i udp@172.16.0.132
COMMAND  PID USER   FD   TYPE DEVICE SIZE NODE NAME
nmbd    4477 root   17u  IPv4 467997       UDP 172.16.0.132:netbios-ns
nmbd    4477 root   18u  IPv4 467998       UDP 172.16.0.132:netbios-dgm

linux:~ # lsof -i @172.16.0.132:ssh     bei Clientzugriff Clientport angeben!
COMMAND   PID USER   FD   TYPE DEVICE SIZE NODE NAME
sshd    15307 root    3u  IPv6 475796       TCP 172.16.0.132:ssh->172.16.0.244:neod1 (ESTABLISHED)
sshd    15466 root    3u  IPv6 482923       TCP 172.16.0.132:ssh->172.16.0.244:neod2 (ESTABLISHED)

linux:~ # lsof -i tcp:ssh
COMMAND   PID USER   FD   TYPE DEVICE SIZE NODE NAME
sshd     4777 root    3u  IPv6   9151       TCP *:ssh (LISTEN)
sshd    15307 root    3u  IPv6 475796       TCP 172.16.0.132:ssh->172.16.0.244:neod1 (ESTABLISHED)
sshd    15466 root    3u  IPv6 482923       TCP 172.16.0.132:ssh->172.16.0.244:neod2 (ESTABLISHED)

linux:~ # lsof -i :1-512
```

Bedeutung der Adress-Kürzel:

[protocol]	tcp oder udp (icmp öffnet keine Dateien)
[@hostname|hostaddr]	IP oder Hostname
[:service|port]	Port oder Dienstname aus /etc/serices

jede dieser Teilangaben kann weggelassen werden - Identifikation durch führendes @ oder :

netcat

Einsatz: **Analyse von TCP/IP-Verbindungen – verbindet seine Standardein- und -ausgabekanäle mit einer Rechner-/Portkombination**
kann sowohl Client als auch Server simulieren
kann sowohl TCP als auch UDP verwenden
Es umgeht somit die Nachteile von telnet als Verbindungstest:

- Es kann Server spielen
- Es kann das UDP-Protokoll verwenden.
- Es kann korrekt mit binären Datenströmen umgehen.
- kein EOF-Problem

nc <Rechner> <Port> **einfachste Aufrufform**
alles, was von stdin gelesen wird, wird auf diesem Netzwerk-Kanal gesendet
alles, was von diesem Kanal empfangen wird, wird auf dem Monitor (stdout) dargestellt

allgemeine Syntax:

als Server (TCP)	**als Client (TCP)**
netcat -l -p <Port>	netcat <Server> <Port>
als Server (UDP)	**als Client (UDP)**
netcat -l -u -p <Port>	netcat -u <Server> <Port>

Übertragung von Text:

Server	Client
server:~ # **netcat -vv -l -p 1234** listening on [any] 1234 ...	
connect to [server] from (UNKNOWN) [client] 32849	client:~ # **netcat -vvv server 1234** (UNKNOWN) [server] 1234 (search-agent) open
etwas text	etwas text

Übertragung einer Datei:

Server	Client
server:~ # **netcat -vv -l -p 1234 < datei** listening on [any] 1234 ...	
connect to [client] from (UNKNOWN) [client] 32850	client:~ # **netcat -vvv server 1234 > datei** (UNKNOWN) [server] 1234 (search-agent) open
sent 433841, rcvd 0	**Ctrl + C** sent 0, rcvd 433841

Protokolle zu Fuß und im Batch-Betrieb nutzen:

```
linux:~ # echo "GET /adressframe/adresse.htm HTTP/1.1
Host: www.zurquelle.de

" | netcat www.zurquelle.de 80
```

Bau eines Mini-Webservers und Abhören der Browseranfrage

Server	Client (Internet Explorer)
server:~ # **netcat -l -p 80**	
GET / HTTP/1.1 Accept: image/gif, image/x-xbitmap, Accept-Language: de Accept-Encoding: gzip, deflate User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1) Host: 172.16.0.132 Connection: Keep-Alive	in Adressleiste: **http://server**
hallo **Ctrl + C** punt!	Ausgabe im Browserfenster: hallo

nmap

Einsatz: **(verschleiertes) Scannen von offenen Ports auf entfernten Rechnern**
verschafft einen Überblick über alle erreichbaren Dienste im LAN oder anderen Netzen

```
linux:~ # nmap --help
Nmap 3.70 Usage: nmap [Scan Type(s)] [Options] <host or net list>
Some Common Scan Types ('*' options require root privileges)
* -sS TCP SYN stealth port scan (default if privileged (root))
  -sT TCP connect() port scan (default for unprivileged users)
* -sU UDP port scan
  -sP ping scan (Find any reachable machines) → wird vor jedem Scan gemacht
* -sF,-sX,-sN Stealth FIN, Xmas, or Null scan (experts only)
  -sV Version scan probes open ports determining service & app names/versions
  -sR RPC scan (use with other scan types)
Some Common Options (none are required, most can be combined):
* -O Use TCP/IP fingerprinting to guess remote operating system
  -p <range> ports to scan.  Example range: 1-1024,1080,6666,31337
  -F Only scans ports listed in nmap-services
  -v Verbose. Its use is recommended.  Use twice for greater effect.
  -P0 Don't ping hosts (needed to scan www.microsoft.com and others)
* -Ddecoy_host1,decoy2[,...] Hide scan using many decoys
  -6 scans via IPv6 rather than IPv4
  -T <Paranoid|Sneaky|Polite|Normal|Aggressive|Insane> General timing policy
  -n/-R Never do DNS resolution/Always resolve [default: sometimes resolve]
  -oN/-oX/-oG <logfile> Output normal/XML/grepable scan logs to <logfile>
  -iL <inputfile> Get targets from file; Use '-' for stdin
* -S <your_IP>/-e <devicename> Specify source address or network interface
  --interactive Go into interactive mode (then press h for help)
Example: nmap -v -sS -O www.my.com 192.168.0.0/16 '192.88-90.*.*'
```

nmap –sT <Zielrechner> **normaler TCP-Connect-Scan mit Default-Optionen**

- dieser Modus ist bei einem normalen Systembenutzer Standard
- gescannt werden die Ports 1 – 1024 sowie alle in /etc/services eingetragene Ports

```
linux:~ # nmap -sT 172.16.0.132
Starting nmap 3.70 ( http://www.insecure.org/nmap/ ) at 2005-01-27 11:30 CET
Interesting ports on 172.16.0.132:
(The 1650 ports scanned but not shown below are in state: closed)
PORT     STATE SERVICE
7/tcp    open  echo
22/tcp   open  ssh
...
```

nmap –sS -P0 <Zielrechner> **Stealth-Scan mit Default-Optionen ohne vorherigen Ping**

- Standard für Benutzer root
- halboffener Scan: sendet SYN, erhält SYN,ACK, sendet sofort RST

nmap –p <Ports> <Zielrechner> Scannen festgelegter Ports

```
linux:~ # nmap -p 20-25,80,110,443 172.16.0.244
Starting nmap 3.70 ( http://www.insecure.org/nmap/ ) at 2005-01-27 11:44 CET
Interesting ports on 172.16.0.244:
PORT    STATE  SERVICE
20/tcp  closed ftp-data
21/tcp  closed ftp
22/tcp  closed ssh
23/tcp  closed telnet
24/tcp  closed priv-mail
25/tcp  closed smtp
80/tcp  closed http
110/tcp closed pop3
443/tcp closed https
```

nmap –sU <Zielrechner> **Scannen von UDP-Ports**

- sendet UDP-Pakete an den entsprechenden Port und wartet auf "port unreachable → Port ist geschlossen
- Achtung: falls eine Firewall alle nicht benutzten Ports abblockt, werden alle UDP-Ports als offen angezeigt!!!

```
linux:~ # nmap -sU -p 53 172.16.0.244
PORT   STATE  SERVICE
53/udp closed domain
```

tcpdump

alternativ: ngrep

Einsatz: **Abhören von Netzwerkverkehr,** nutzt Bibliothek libpcap

```
linux:~ # tcpdump --help
Usage: tcpdump [-aAdDeflLnNOpqRStuUvxX] [-c count] [ -C file_size ]
               [ -E algo:secret ] [ -F file ] [ -i interface ] [ -r file ]
               [ -s snaplen ] [ -T type ] [ -w file ] [ -y datalinktype ]
               [ expression ]
```

Filterausdrücke

udp	zeigt UDP-Pakete	broadcast	zeigt Broadcastverkehr
tcp	zeigt TCP-Pakete	host H	zeigt Pakete von oder zu Rechner H
icmp	zeigt ICMP-Pakete	src host	bezieht host nur auf die Quelle
arp	zeigt ARP-Pakete	dst host	bezieht host auf das Ziel
src port	bezieht port nur auf die Quelle	dst port	bezieht port auf das Ziel
net N/B	zeigt Verkehr innerhalb, von oder zu Netzwerkadresse/Bitanzahl		
port P	zeigt TCP/UDP-Pakete von oder zu Port oder Dienst P		
Verknüpfungen			
A and B	A und B müssen zutreffen	not A	negiert den Filterausdruck A
A or B	A oder B muss zutreffen	(A)	klammert A als Teilausdruck

tcpdump –n udp Abhören von UDP-Paketen ohne Namensauflösung:

```
linux:~ # tcpdump -n udp
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 96 bytes
12:02:12.585790 IP 172.16.0.60.137 > 172.16.0.255.137: NBT UDP PACKET(137): QUERY;
REQUEST; BROADCAST
```

Ausgabeformat:

```
<Zeitstempel> <IP mit Quellport> <IP mit Zielport> <variabler Datenteil> <Flags>
```

tcpdump –vvv sehr ausführliche Ausgabe (abhängig vom Protokoll)

```
linux:~ # tcpdump -n udp -vvv
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 96 bytes
12:21:39.384953 IP (tos 0x0, ttl 128, id 13265, offset 0, flags [none], length: 229)
172.16.0.106.138 > 172.16.0.255.138:
>>> NBT UDP PACKET(138) Res=0x1102 ID=0x804B IP=172 (0xac).16 (0x10).0 (0x0).106
(0x6a) Port=138 (0x8a) Length=187 (0xbb) Res2=0x0
SourceName=FM24W2K-OLI     NameType=0x20 (Server)
DestName=
WARNING: Short packet. Try increasing the snap length
```

tcpdump –n arp -i ppp0 Abhören der Einwahlverbindung nach ARP-Paketen ohne Namensauflösung

tcpdump –e arp Anzeige von Ethernetadressen:

```
linux:~ # tcpdump -n arp -e
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 96 bytes
12:16:40.930801 00:4f:4e:0e:73:49 > ff:ff:ff:ff:ff:ff, ethertype ARP (0x0806), length
60: arp who-has 172.16.0.7 tell 172.16.0.2
```

tcpdump –c 100 –i eth2 Abfangen von genau 100 Paketen von Device eth2

tcpdump –evvvxX –s 0 Inhalt der Datenpakete in ASCII und Hex anzeigen lassen

```
12:26:49.144139 00:02:3f:db:4c:23 > 00:11:2f:be:59:51, ethertype IPv4 (0x0800), length
60: IP (tos 0x0, ttl 128, id 45080, offset 0, flags [DF], length: 40)
172.16.0.244.ansyslmd > 172.16.0.132.ssh: . [tcp sum ok] 445173:445173(0) ack 4340360
win 60683
        0x0000:  0011 2fbe 5951 0002 3fdb 4c23 0800 4500  ../.YQ..?.L#..E.
        0x0010:  0028 b018 4000 8006 f11e ac10 00f4 ac10  .(..@...........
        0x0020:  0084 041f 0016 be3f 3123 cba2 c224 5010  .......?1#...$P.
        0x0030:  ed0b e7d0 0000 0000 0000 0000            ............
```

tcdump (port http or port https) and host <Zielrechner> Webaufrufe mitschneiden
tcpdump 'udp and host (host1 or host2)' Netzverkehr von 2 Rechnern
tcpdump not host 10.10.0.7 –w > rawformat-datei Netzverkehr von host1 ausblenden, Ausg. in Datei

nslookup

```
Usage:  nslookup [ -option ]  [ name | - ]  [ server ]
```

nslookup **Start im interaktiven Modus**

```
      set type=A          Anfragen beziehen sich auf A-Records
      set type=PTR
      set type=HINFO
      MX                  SMTP-Mailserver
      exit                nslookup beenden

interaktiv
      nslookup
      ls –a www.zurquelle.de
Kommandozeile
      nslookup –querytype=SOA www.zurquelle.de

nslookup                          Nameserver von t-online erfragen
      set type=NS
      www.t-online.de             liefert FQDN des Nameservers zurück
      set type=A
      <Eingabe des FQDN>          Erfragen der zugehörigen IP
      exit
```

host

```
Usage: host [-aCdlrTwv] [-c class] [-n] [-N ndots] [-t type] [-W time]
            [-R number] hostname [server]
```

host <Zielhost> **Namensauflösung des Zielhosts (NS aus resolv.conf)**

```
linux:~ # host www.zurquelle.de
www.zurquelle.de has address 192.67.198.3
```

host <Zielhost> <Nameserver> Namensauflösung des Zielhosts (NS lt. Parameter 2)

```
linux:~ # host www.zurquelle.de 217.237.149.225
Using domain server:
Name: 217.237.149.225
Address: 217.237.149.225#53
Aliases:
www.zurquelle.de has address 192.67.198.3
```

host –l **alle Einträge einer Domain listen**

dig

```
Usage:  dig [@global-server] [domain] [q-type] [q-class] {q-opt}
        {global-d-opt} host [@local-server] {local-d-opt}
        [ host [@local-server] {local-d-opt} [...]]
```

dig FQDN <Record> **Suche nach dem Record**

dig notebook.example.org A Suche nach der IP von notebook.example.org

```
      ;; ANSWER SECTION:
      notebook.example.org.   7200  IN   A    172.16.0.198
```

dig @ns2.example.org notebook.example.org A w. o., aber Anfrage geht gezielt an den Nameserver ns2

dig 198.0.16.172.in-addr.arpa PTR Reverse-Abfrage nach dem FQDN dieser IP

```
      ;; ANSWER SECTION:
      198.0.16.172.in-addr.arpa. 7200 IN   PTR   notebook.example.org.
```

dig –t MX www.zurquelle.de **Abfrage des MX-Records einer Domain**

notebookneu:~ # dig +trace www.zurquelle.de **listet den Weg der Namensauflösung von den root-Servern ausgehend**

notebookneu:~ # dig +trace www.zurquelle.de @217.237.149.225 umgeht Problem des lokalen DNS-Cache

```
ifstatus eth0

 ethtool
Weitere Parameter mit ethtool -s eth0 speed 100 duplex half autoneg on
```

```
Wir legen zwei neue Routingtabellen an:
root@server:~ # ip route add 192.168.0.0/24 table 100 dev eth0 src 192.168.0.1
protocol static
root@server:~ # ip route add 192.168.0.0/24 table 101 dev eth1 src 192.168.0.2
protocol static


Tabelle 100 routet jetzt nun alles über die eigene IP-Adresse von der Karte
192.168.0.1 (eth0) und Tabelle 101 routet alles über sich selber auf 192.168.0.2
(eth1)
Wir haben jetzt eigentlich nichts weiter gesagt das die beiden IP-Adressen über
jeweils seine eigene Netzwerkkarte routen soll.
Jetzt sagen wir den Kernel welche Tabelle bei welcher source Client IP-Adresse er
verwenden soll.

root@server:~ # ip rule add to 192.168.0.3 table 100
root@server:~ # ip rule add to 192.168.0.4 table 101


Zum Schluss der Konfiguration sollte man immer den Routing Cache löschen

root@server:~ # ip route flush cache
Mit dem ip Programm lassen sich alle Tabellen und Regeln auch ausgeben.
Die Liste der Routingtabellen müsste jetzt so beginnen:

root@server:~ # ip route list table all
192.168.0.0/24 dev eth0 table 100 proto static scope link src 192.168.0.1
192.168.0.0/24 dev eth1 table 101 proto static scope link src 192.168.0.2
192.168.0.0/24 dev eth0 scope link
127.0.0.0/8 dev lo scope link
[....]


Die Regelliste müsste schliesslich noch dies enthalten:

root@server:~ # ip rule list
0: from all lookup local
32764: from all to 192.168.0.3 lookup 101
32765: from all to 192.168.0.4 lookup 100
32766: from all lookup main
32767: from all lookup default
```