Einführung Unix/Linux

Kapitel: Netzwerke

Jens Roesen <jens@roesen.org>

Würzburg, März 2010
Version: 0.1.5 – **sehr beta** –

Dieses Dokument wurde in vim (`http://www.vim.org`) bzw. TEXnicCenter (`http://www.texniccenter.org/`) geschrieben und mit LaTeX (`http://www.latex-project.org/`) formatiert und gesetzt.
Die jeweils aktuelle Version ist unter `http://www.roesen.org` erhältlich.

Inhaltsverzeichnis

Vorwort

Motivation

Im Rahmen interner Schulungsmassnahmen kam die Frage nach geeigneten Schukungsmaterialien bzw. einem Skript für Unix-Neulinge auf. Schulungsunterlagen und Skripte für Einsteiger, aber letztendlich hat mir bei den meisten entweder etwas gefehlt, oder es war für unseren Zweck viel zu viel irrelevanter Stoff. Da wir in der Hauptsache mit Solaris und Linux-Systemen arbeiten und dabei Themen wie X-Windows oder Druckerverwaltung komplett ausklammern können, aber auf Themen wie Netzwerke, Troubleshooting, Mailserver oder DNS-Server Wert legen, habe ich mich irgendwann hingesetzt und angefangen dieses Skript zu schreiben.

Es ist der Versuch Systemadministratoren mit Grundkenntnissen in Unix den Arbeitsalltag zu erleichtern und dabei zwei verschiedene Unix-Geschmacksrichtungen, nämlich Solaris[1] und Red Hat Enterprise Linux, gleichermassen zu betrachten.

Mir ist durchaus klar, dass nicht alles im Folgenden beschriebene „state of the art“ ist bzw. sein kann und sicher auch noch etliche Fehler übersehen wurden. Wer einen solchen findet, weiss wie man einige Aufgaben besser lösen kann oder bessere Beispiele kennt ist herzlich eingeladen mir eine Mail an <jens@roesen.org> zu schicken.

Zielgruppe

Dieses Kurzscript ist als Crashkurs zur Einführung in die Administration von Unix und Linux Systemen gedacht. Es wird dabei ausschliesslich auf der Konsole und ohne grafische Oberfläche gearbeitet. Die gezeigten Beispiele beziehen sich auf Systeme unter Sun Solaris und RedHat Enterprise Linux.

Ohne Vorkenntnisse und Erfahrung mit Unix und/oder Linux Systemen wird der angesprochene Stoff teils nur schwer zu verstehen sein. Als alleiniges Lehrskript für blutige Anfänger ist es daher nicht geeignet obwohl in einigen Kapiteln vereinzelt kurz auf Grundlagen eingegangen wird (z.B. Kapitel 2).

Aufbau des Skripts

Wirr. Durch und durch. Aber zu mehr ist momentan keine Zeit. Ich habe versucht die Kapitel und Themen in eine halbwegs sinnvolle Reihenfolge zu bringen. Im Lauf der Zeit wird da sicherlich noch einiges umgestellt werden.

[1] In der vorliegenden Version des Skripts nur bis Version 9.

Typographisches

Da sich alle Beispiele, Kommandos und Ausgaben auf der Konsole abspielen, werden diese Bereiche entsprechend formatiert um sich vom regulären Text abzusetzen. Nach dem Login als User root, mit dem wir in diesem Skript hauptsächlich arbeiten werden, landet man in einem Command Prompt der so aussehen koennte:

```
[root@server1 root]#
```

Da dieser Prompt ja nach name des Systems oder aktuellem Verzeichnis mal kürzer aber auch sehr viel länger sein kann, wird der root Prompt auf

```
#
```

verkuerzt. Bitte den Hash (`#`) hier **nicht** als Kommentarcharakter verstehen, der unter Linux/Unix z.B. in Shellskripten die folgende Zeile vor der Ausführung durch die Shell schützt. Der normale User-Prompt, falls er uns wirklich einmal begegnen sollte, wird analog dazu auf

```
$
```

zusammengestrichen.

Für Konsolenausgaben, Konfigurationsdateien oder Teile von Skripten wird eine nichtproportionale Schrift verwendet:

```
if [ -n "$_INIT_NET_IF" -a "$_INIT_NET_STRATEGY" = "dhcp" ]; then
        /sbin/dhcpagent -a
fi
```

Werden in einem Beispiel Konsoleneingaben vom Benutzer erwartet, wird die in einer nichtproportionale Schrift dargestellt, wobei die Benutzereingaben fett gedruckt sind:

```
# uname -a
SunOS zoidberg 5.9 Generic_118558-21 sun4u sparc SUNW,Sun-Blade-100
```

Kommandos, Dateinamen oder Benutzerkennungen im laufenden Text werden ebenfalls in einer nichtproportionalen Schrift dargestellt: „Mit dem Befehl `pwd` kann überprüft werden, in welchem Verzeichnis man sich gerade befindet.“.

Müssen in einem Beispiel noch Teile der erwarteten Benutzereingaben durch die richtigen Werte ersetzt werden, so wird dieser Teil in kursiver Nichproportionalschrift dargestellt: „Für jedes Interface welches beim boot konfiguriert werden soll muß eine Datei **/etc/hostname.*interface*** existieren.“.

Eigennamen, Personen oder Organisationen erscheinen manchmal (ich bin gerade zu faul alle Vorkommen entsprechend zu formatieren) in Kapitälchen: „Eine sehr große Rolle hierbei hat die University of California in Berkley (UCB) gespielt, an der Thompson im Winter 76/77 eine Vorlesung zum Thema Unix abhielt.“.

1 Netzwerke

> A name indicates what we seek.
> An address indicates where it is.
> A route indicates how we get there.
>
> *(Jon Postel, RFC 791)*

In diesem Kapitel werden wir kurz das OSI Modell[1], das Adress Resolution Protocol[2] und die IP Adressierung wiederholen, bevor wir uns der Konfiguration von Netzwerkinterfaces widmen. Dabei betrachten wir nur IPv4[3] und lassen IPv6[4] außen vor.

1.1 Das Open Systems Interconnect Modell

Bereits 1979 begann die Entwicklung eines mehrschichtigen Referenzmodells mit dessen Hilfe Kommunikationsprotokolle beschrieben und entwickelt werden können. Dieses Modell regelt welche Anforderungen die im einzelnen beteiligten Protokolle erfüllen müssen um eine reibungslose Kommunikation zu gewährleisten. Die Standardisierung des Referenzmodells erfolgte 1983.

Das OSI Modell besteht aus sieben Schichten von Diensten die eine Grundlage für Netzwerkommunikation und Kommunikationsprotokolle bilden. Jede Schicht (layer) muss bei der Kommunikation die ihr zugedachten Aufgaben erfüllen und reicht danach die bearbeiteten Daten an die nächst niedrigere (beim senden) bzw. nächst höhere (beim empfangen) Schicht weiter. Jede Schickt verpackt die Daten zusätzlich indem sie einen Header und einen Footer hinzufügt. Dieser Vorgang wird data encapsulation genannt und ermöglichet die Kommunikation der einzelnen Layer untereinander.

Abbildung 1.1 auf Seite 2 zeigt den Aufbau des Modell mit einigen bekannten Protokoll- und Servicebeispielen.

1.1.1 Application Layer / Anwendungsschicht

Der Application layer stellt als oberste Schicht des OSI Modells den darüberliegenden Anwendung eine Schnitstelle zum Netzwerk dar. Dies können Browser, Mail Clients und

[1] Durch die Internationale Organisation für Normung (ISO) im Jahrt 1983 standardisiert.
[2] RFC 826 http://tools.ietf.org/html/rfc826
[3] RFC 791 http://tools.ietf.org/html/rfc791
[4] Unter anderem RFC 2460 http://tools.ietf.org/html/rfc2460

Open Systems Interconnect Modell				
OSI Schicht		Aufgabe	Protokoll	Einheit
7	Application	Anwendungsorientiert	HTTP, FTP, SMTP	Daten
6	Presentation			
5	Session			
4	Transport	Transportorientiert	TCP UDP	Segmente
3	Network		IP ICMP IGMP	Pakete
2	Data Link		Ethernet, Token Ring	Frames
1	Physical			Bits

Abbildung 1.1: OSI Schichtenmodell

ähnliches sein.

1.1.2 Presentation Layer / Darstellungsschicht

Diese Schicht setzt die verschiedenen Datenformate beteiligter Systeme in eine einheitliche Form um und stellt sicher, dass die Daten von der Anwendungsschicht des Kommunikationspartners auch gelesen werden können.

1.1.3 Session Layer / Sitzungsschicht

Diese schickt kümmert sich um die Verbindung, session, und die Prozesskommunikation beteiligter Systeme.

1.1.4 Transport Layer / Transportschicht

Die Transportschicht stellt eine Ende-zu-Ende Kommunikation für Sender und Empfänger zur Verfügung und sorgt dafür, dass die Daten auch beim Empfänger ankommen.

1.1.5 Network Layer / Vermittlungsschicht

Die Vermittlungsschicht kümmert sich um die Weiterleitung der Datenpakete, stellt die Adressierung (z.B. IP) sicher und verwaltet die Routingtabellen.

1.1.6 Data Link Layer / Sicherungsschicht

Die Sicherungsschicht soll eine zuverlässige und fehlerfreie Kommunikation durch die unter ihr liegenden Netzwerkschichten gewährleisten. Die übetragenden Frames enthalten eine Prüfsumme und können so bei Bedarf neu angefordert werden. Die Sicherungsschicht ist in zwei Unterschichten geteilt: die obere namens Logical Link Control (LLC) und darunter die Media Access Control (MAC).

1.1.7 Physical Layer / Bitübertragungsschicht

Die unterste Schickt des OSI Modells definiert die zugrundeliegende Hardware und die Art der Signale welche für die Datenübertragung genutzt werden.

1.2 TCP/IP Referenzmodell

Das TCP/IP Modell ist wie das OSI Referenzmodell in Schichten aufgebaut. Es baut auf dem vom US Verteidigungsministerium 1970 entwickelten DoD-Schichtenmodell auf.

Im Gegensatz zum OSI Modell besteht das TCP/IP Modell aus nur vier Schichten, die sich aber auf im OSI Modell wiederfinden. Abbildung 1.2 beschreibt die vier Schichten und nennt die vergleichbaren OSI Schichten.

TCP/IP Schichtenmodell		
TCP/IP-Schicht	≡ OSI-Schicht	Protokoll Beispiel
Application layer	5-7	HTTP, FTP, SMTP
Transport Layer	4	TCP, UDP, SPX
Internet Layer	3	IPv4, IPv6
Link Layer	1-2	Ethernet, FDDI, Token Ring

Abbildung 1.2: TCP/IP Schichtenmodell

1.2.1 Application layer / Anwendungsschicht

Die Anwendungsschicht beinhaltet Protokolle, die entsprechenden Anwendungen Dienst zur Verfügung stellen.

1.2.2 Transport Layer / Transportschicht

Das wichtigste Protokoll der Transportschicht, das Transmission Control Protocol (TCP), stellt eine End-zu-End-Verbindung her. Es arbeitet verbindungsorientiert und soll so eine zuverlässige Datenübertragung ohne Paketverluste garantieren. Verbindungen werden über einen three-way-handshake (`SYN`, `SYN,ACK`, `ACK,DATA`) hergestellt. Zu dieser Schicht gehoren aber auch „unzuverlässige“ Protokolle wie das verbindungslose User datagram Protocol (UDP).

1.2.3 Internet Layer / Internetschicht

Die Internetschicht kümmert sich um das Routing und die Vermittlung der Segmente und Pakete höherliegender Schichten als Datagramm zum nächsten Wegpunkt/Hop. Sie ist außerdem für die Adressierung der beteiligten Systeme sowie für Fragmentierung und Defragmentierung der Datagramme verantwortlich.

1.2.4 Link layer / Netzzugangsschicht

Die Netzzugangsschicht des TCP/IP-Models beinhaltet keine eigenen Protokolle, sondern verlässt sich auf die unteren beiden Schichten des OSI-Modells.

1.3 IP Adressierung und Subnetting

Im Grunde setze ich voraus, daß jeder der dieses Skript liest wenigstens in Ansätzen die IP-Adressierung verstanden hat. Wer noch völlig unbefleckt auf diesem Gebiet ist soll sich entweder im Internet schlau machen oder das Buch „TCP/IP Netzwerkadministration“ aus dem O'Reilly Verlag (ISBN 3897211793) besorgen. Ich werden das Thema, genauer gesagt die IPv4 Adressierung, daher nur kurz zur Wiederholung anschneiden.

1.3.1 IP-Adressen und Subnetze

IP-Adressen sind 32 Bit lang und werden in der Regel als „dotted decimal“ notiert: vier ganze Dezimalzahlen zwischen 0 und 255 und je durch einen Punkt getrennt. Jede Zahl stellt ein Oktett bzw. Byte dar. Hier im Vergleich die IP-Adresse 192.168.2.28 einmal binär und dezimal:

$$\underbrace{11000000}_{192.}.\underbrace{10101000}_{168.}.\underbrace{00000010}_{2.}.\underbrace{00011100}_{28}$$

Das erste Oktett aus den 8 Bits 11000000 entspricht dezimal also dem Wert 192. Wieso? Jedes der 8 Bits entspricht einem Wert. Von links nach rechts[5]: 128 64 32 16 8 4 2 1. Sind nun die ersten beiden Bits je 1 und alle folgenden Bits 0 muß man für den Dezimalwert die Werte der ersten beiden Bits addieren. 128+64=192. Genauso für das zweite Oktett 10101000: 128+32+8=168. Und so weiter. Eigentlich ganz einfach.

Zu jeder IP-Adresse gehoert auch eine Subnetzmaske welche die Größe des Netzwers in dem sich unsere IP befindet bestimmt. Diese Subnetzmaske teilt die IP in einen unveränderlichen Netzwerk- und einen veränderlichen Hostteil. Subnetzmasken können entweder ebenfalls dottet decimal oder als CIDR-Suffix gaschrieben werden. CIDR[6], Classless Inter-Domain Routing, wurde eingeführt um die IP-Adressräume effizienter zu nutzen und Routingtabellen verkleinern zu können. Vor CIDR gab es drei Subnetzgrößen. Class-A Netz (Subnetzmaske 255.0.0.0) mit 16777214 Hosts pro Netz, Class-B (255.255.0.0) mit 65534 Hosts pro Netz und schliesslich Class-C (255.255.255.0) mit 254 Hosts pro Netz. Wenn Example Corp. nun also ein offizielles Netz mit 600 Hosts gebrauchte hätte, wäre ein Class-C Netz zu klein und ein Class-B Netz eine gigantische Verschwendung von Adressen gewesen. Natürlich hätte man Example Corp. entsprechend viele Class-C Netze zuweisen können, dadurch wären aber die Routingtabellen weiter angewachsen.

[5] Ja, es wird binär von rechts nach links, von klein nach groß, gelesen, aber das finde ich persönlich beim Thema IP und Subnetze nur verwirrender.

[6] RFC 1518 (http://tools.ietf.org/html/rfc1518) und RFC 1519 (http://tools.ietf.org/html/rfc1519) von 1993

Ein Subnetzmaske bei der man früher von einem Class-C Netz gesprochen hat lässt sich also dotted decimal als 255.255.255.0 oder nach CIDR als Suffix /24 an die IP-Adresse anfügen: 192.168.2.28/24. Warum die 24? Weil bei einer Netzmaske von 255.255.255.0 die ersten 3 Oktette oder eben die ersten 24 Bits auf 1 stehen.

$$\underbrace{11111111}_{255.}.\underbrace{11111111}_{255.}.\underbrace{11111111}_{255.}.\underbrace{00000000}_{0}$$

Der Netzanteil der IP-Adresse ist, wie vorhin schon erwähnt, unveränderlich. Das heisst die IP-Adresse alle Hosts aus diesem Netz (192.168.2.0/24) beginnen mit 192.168.2. und unterscheiden sich erst in der letzten Zahl der IP, dem Hostteil.

$$\underbrace{192\ .\ 168\ .\ 2\ .}_{Netzwerkteil}\ \underbrace{28}_{Hostteil}$$

In jedem Netz mit n IP-Adressen koennen maximal n-2 Hosts existieren, denn die erste IP des Netzes ist fuer die Netzwerdadresse und die letzte IP fuer die Broadcastadresse reserviert[7]. In unserem Beispiel sind von den 256 möglichen IPs (192.168.2.0 bis 192.168.2.255) 254 für Hosts nutzbar.

1.3.2 Subnetting

Da Subnetting vielen angehenden (und teils auch gestandenen Admins) ein Dorn im Auge ist werde ich das Thema auch nicht herumkommen. Solltest du Subnetting mit links und verbundenen Augen beherrschen überspring dieses Teil, denn ich verwende letztendlich eine etwas andere Art der Berechnung als die „offiziell“ propagierte und will mögliche Verwirrung vermeiden.

In den meisten Lehrbüchern wird auf die Formel $2^n - 2$ bzw. 2^n verwiesen. Wenn man ein Subnetz mit Suffix /27 hat sind 27 Bits für den Netteil und die übrigen 5 Bits $(32-27 = 5)$ für den Hostteil vorgesehen. Pro Subnet können $2^5-2 = 30$ Hosts existieren. Mit Betrachtung auf ein Class-C Netz mit Suffix /24 verwendet unser Subnetz mit Suffix /27 zusätzliche 3 Bits für die Netzmaske. Es kann also $2^3-2 = 6$ bzw., da wir anständige Hardware besitzen die `ip subnet-zero`[8] verstehen, $2^3 = 8$ Subnetze. Auf die Weise, mit Bits zählen und Potenzen bilden, kommt man prima durch jedes Subnetting. Ich persönlich hab das zwar verstanden, fand es aber nicht sonderlich anschaulich und hab in all den Jahren so manchen Auszubildenden und CCNA-Anwärter daran verzweifeln sehen. Also: $2^n - 2$ und 2^n my ass!

Bleiben wir bei dem Beispiel von oben: ein Netz mit eine Netzmaske /27. Die 3 zusätzlichen Bytes im Vergleich zu /24 bewirken eine in dotted decimal geschriebene Netzmaske von 255.255.255.224. Von den 256 IPs (0 bis 255) eines /24 Netzes ziehe ich die 224 der /27 Netzmaske ab und ich komme auf meine 32 IPs pro Subnetz. Abzüglich

[7] Theoretisch kann die Netzadresse auch frei benutzt werden, solang sich keine Windows 9x Hosts im Netz befinden. Aber laut sagen darf man das eigentlich nicht. Jehova!

[8] Früher sollten das erste (all-zeros) und letzte (all-ones) Subnet nicht benutzt werden. Heute ist das allerdings kein Problem mehr. Siehe auch `http://www.cisco.com/application/pdf/paws/13711/40.pdf`

Netzwerkadresse und Broadcast bleiben 30 IPs fuer Hosts. Ich kann 256/32 = 8 /27 Subnetze bilden. Wer ungern teilt kann sich auch die Reihe „128 64 32 16 8 4 2“ vorstellen und dann einfach „an der 16 spiegeln“: bei 32 IPs gibt es 8 Netze, bei 64 Netzen nur 4 IPs pro Netz usw.. Klingt komisch, funktioniert aber auch. Supernetting ist so auch möglich.

Letztendlich gibt es verschiedene Möglichkeiten Subnetting im Kopf zu betreiben. Wichtig ist nur, das man es verstanden hat und seine Auswirkungen versteht. Zur Not gibt es ja auch noch `ipcalc`[9].

```
# ipcalc -n 192.168.2.1/27
Address:   192.168.2.1          11000000.10101000.00000010.000 00001
Netmask:   255.255.255.224 = 27 11111111.11111111.11111111.111 00000
Wildcard:  0.0.0.31             00000000.00000000.00000000.000 11111
=>
Network:   192.168.2.0/27       11000000.10101000.00000010.000 00000
HostMin:   192.168.2.1          11000000.10101000.00000010.000 00001
HostMax:   192.168.2.30         11000000.10101000.00000010.000 11110
Broadcast: 192.168.2.31         11000000.10101000.00000010.000 11111
Hosts/Net: 30                    Class C, Private Internet
```

1.4 ARP - Address Resolution Protocol

Das Address Resolution Protocol ist ein Protokoll der Netzzugangsschicht des TCP/IP Modells und ermittelt die zu einer IP-Adresse zugehörige 48Bit lange Hardwareadresse, auch MAC-Adresse, eines Netzwerkgeräts im lokalen Netz. Falls die MAC-Adresse eines Zielrechners noch nicht in der ARP-Tabelle ist, schickt der Host einen ARP-Request als Paket mit der Destination-IP des gesuchten Rechners per ARP-Broadcast (`ff-ff-ff-ff-ff-ff`) an alle Rechner im Lan. Findet ein Rechner seine eigene IP als Empfänger in dem ARP-Request antwortet er dem Absender mit seiner MAC-Adresse.

```
14:34:22.611372 arp who-has server-b.example.com tell server-a.example.com
14:34:22.611641 arp reply server-b.example.com is-at 00:03:ba:09:bf:51
```

Ist der Empfänger eines IP-Pakets im lokalen Netz, werden alle Pakete für diesen Empfänger direkt an seine MAC-Adresse gesendet. Liegt der Empfänger ausserhalb des LAN werden die Pakete an die MAC-Adresse des entsprechenden Gateways gesendet, die Destination-IP wird dabei nicht geändert.

Die ARP-Tabelle kann mit arp (Linux) bzw. arp -a (Solaris, unter Linux Ausgabe im BSD-Stil) abgefragt und bei Bedarf manipuliert werden. Der Switch -n unterdrückt die Namensauflösung.

```
# arp -a
Net to Media Table: IPv4
Device   IP Address               Mask      Flags   Phys Addr
------ -------------------- --------------- ----- ---------------
```

[9] Online Tool und Download unter `http://jodies.de/ipcalc`

```
eri0   torwaechter          255.255.255.255       00:14:4f:1f:b1:eb
eri0   subint               255.255.255.255 SP    00:03:ba:24:92:a9
eri0   oeko-pix             255.255.255.255       00:0b:46:22:25:91
eri0   gandhi               255.255.255.255       00:03:ba:6c:e4:0d
eri0   pix                  255.255.255.255       00:03:6b:f6:70:e6
eri0   zoidberg             255.255.255.255 SP    00:03:ba:25:6c:a9
eri0   BASE-ADDRESS.MCAST.NET 240.0.0.0       SM    01:00:5e:00:00:00
```

Wie lange Einträge in der ARP-Tabelle bestehen bleiben ist unterschiedlich. Unter Solaris liegt das `arp_cleanup_interval` von `/dev/arp` normalerweise bei 300000 Millisekunden (5 Minuten). Unter Linux liegt der Defaultwert für die `gc_stale_time`[10] bei 60 Sekunden. Notfalls muß ein Eintrag mit `arp -d <`*ipadresse*`>` manuell aus der Tabelle gelöscht werden.

1.5 Routing

Wenn man einen Host erreichen will, der nicht im eigenen (Sub)netz liegt, müssen die Pakete an jemanden geschickt werden, der entweder den Weg zum Ziel kennt oder zumindest eine Ahnung hat in welche Richtung es geht. Das ist der Router. Während Hubs nur auf Layer 1 und Switche nur auf Layer 2 des OSI-Modells arbeiten und daher Pakete nur im lokalen Netz verteilen, arbeitet ein Router auf Layer 3 und ist in mehreren Netzen beheimatet zwischen denen er die Pakete weiterleiten kann.

Um also Daten in entferne Netze zu schicken, muss ein Host zumindest einen Router kennen, an den er diese Pakete schicken kann. Das ist der Defaultrouter. Der Host adressiert die Daten an die IP-Adresse des Zielhosts, schickt sie aber, wie im letzten Abschnitt schon erwähnt, an die MAC-Adresse vom Router. Dieser sucht in seiner Routing Tabelle nach dem besten Weg zum Ziel, im Zweifel ist das der Defaultrouter des Routers, und leitet sie entsprechend weiter. Das geht solange bis das Paket das Ziel erreicht hat oder aus irgendeinem Grund nicht zugestellt werden kann[11]. Die Ziel IP wird auf all diesen Wegen nicht geändert[12]. Dieser Vorgang des Entscheiden über den besten Weg und das anschließende Weiterreichen der Pakete von einer Zwischenstation, einem Hop, zum nächsten wird „Forwarding“ genannt. „Routing“ bezeichnet ansich den gesamten Weg durch das Netz bis zum Ziel.

Das Defaultgateway wird unter Solaris über die Datei `/etc/defaultrouter` vorgegeben. Sie entält pro Zeile die IP oder den Hostnamen[13] eines oder mehrerer Defaultgateways. Unter Linux wird der Defaultrouter in der Datei `/etc/sysconfig/network`[14] über den Parameter `GATEWAY=<`*gateway-IP*`>` vorgegeben.

Die Routing Tabelle eines Hosts kann man sich mit dem Befehl `netstat` und den Optionen `-rvn` anzeigen lassen. Dabei bedeutet das `-rvn` das die Ausgabe der Routing-

[10] Dieser Parameter entspricht nicht dem absoluten ARP-Timeout.

[11] In dem Fall wird der letzte beteiligte Router wahrscheinlich eine Internet Control Message Protocol (ICMP) Nachricht wie „Destination Unreachable“ an den Absender schicken

[12] Solange keine Network Address Translation (NAT) beteiligt ist.

[13] Dieser muss dann über die `/etc/hosts` auflösbar sein.

[14] Siehe auch Kapitel 1.6.2

tabelle ausführlich, verbose, und ohne Namensauflösung erfolgen soll. Folgendes Beispiel zeigt die Routingtabelle eines Solaris Hosts mit einem Interface `eri0` mit der IP 192.168.2.200/27. Das Defaultgateway ist die 192.168.2.193. Als Gateway für das Subnetz 192.168.2.192/27 in dem sich der Host befindet ist die eigene IP angegeben, da dieses Netz direkt erreicht werden kann. Directly connected, wie der Cisco sagt.

```
# netstat -rvn

IRE Table: IPv4
  Destination        Mask           Gateway     Device Mxfrg  Rtt  Ref Flg  Out  In/Fwd
--------------- --------------- --------------- ------ ----- ----- --- --- ----- ------
192.168.2.192   255.255.255.224 192.168.2.200   eri0    1500*    0   1 U    22632      0
224.0.0.0       240.0.0.0       192.168.2.200   eri0    1500*    0   1 U        0      0
default         0.0.0.0         192.168.2.193           1500*    0   1 UG  224238      0
127.0.0.1       255.255.255.255 127.0.0.1       lo0     8232*    0   2 UH      38      0
```

Unter Linux tut es auch schon ein einfaches `route` zum Abfragen der Tabelle.´

```
# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.3.80    *               255.255.255.240 U     0      0        0 eth0
169.254.0.0     *               255.255.0.0     U     0      0        0 eth0
default         install-pc      0.0.0.0         UG    0      0        0 eth0
```

Die Flags (Flg) haben folgende Bedeutung: `U` - Route ist up, `H` - Hostroute und `G` - Route zeigt auf ein Gateway.

Das Solaris `route` bietet mit dem „sub-command“ (betrachtes wir es als Option oder Switch) `get` einen praktischen Weg, um auf Solaris Hosts die über mehrere Netzinterfaces verfügen oder gar selber routen[15] die Route zu einem Ziel zu erfragen.

```
# route get www.example.com
   route to: www.example.com
destination: default
       mask: default
    gateway: 192.168.2.193
  interface: eri0
      flags: <UP,GATEWAY,DONE,STATIC>
 recvpipe  sendpipe  ssthresh    rtt,ms rttvar,ms  hopcount      mtu     expire
       0         0         0         0         0         0      1500          0
```

1.6 Interfacekonfiguration

Viele der grundsätzlichen Einstellungen eines Interfaces werden sowohl unter Solaris als auch unter Linux mit ifconfig vorgenommen bzw. abgefragt. Mit ifconfig -a werden alle

[15] Erkennt das init-Script `/etc/rc2.d/S69inet` beim starten mehr als zwei Netzwerkinterfaces wird IP-Forwadring automatisch aktiviert. Bei Bedarf kann es mit `/usr/sbin/ndd -set /dev/ip ip_forwarding 0` wieder deaktiviert werden.

konfigurierten Interfaces abgefragt und die momentan aktiven Einstellungen ausgegeben. Einzelne Interfaces lassen sich gezielt mit `infconfig` *`interface`* abfragen. Dabei ähneln sich die Ausgaben von Solaris

```
# ifconfig hme0
hme0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 192.168.200.154 netmask fffffffc broadcast 192.168.200.155
        ether 8:0:20:a2:8d:d6
```

und Linux

```
# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:0A:E4:7E:4B:8A
          inet addr:192.168.178.178  Bcast:192.168.178.183  Mask:255.255.255.248
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:120870162 errors:0 dropped:0 overruns:0 frame:0
          TX packets:114609847 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:3045188142 (2904.1 Mb)  TX bytes:2423390401 (2311.1 Mb)
          Base address:0x2400 Memory:dd220000-dd240000
```

Diese Einstellungen können mit **ifconfig** auch geändert werden. Ein **ifconfig hme0 10.1.2.108 mask 255.255.255.128** wuerde z.B. dem Interface **hme0** die IP 10.1.2.108 in einem 25-Bit Subnetz zuweisen. Diese Änderung würde einen reboot der Maschine allerdings nicht überstehen. Solaris und die diversen Linux Distributionen gehen da verschiedene Wege.

1.6.1 Interfacekonfiguration unter Solaris

Um ein Interface bereits beim booten richtig ins das System einzubinden müssen unter Solaris einige Dateien konfiguriert werden. Für jedes Interface welches beim boot konfiguriert werden soll muß eine Datei `/etc/hostname.`*`interface`* existieren. In dieser Datei existiert nur ein Eintrag: entweder der Hostname (wird in die IP aufgelöst) oder die IP-Adresse des Interfaces. Gegebenenfalls muss in der Datei `/etc/netmasks` noch die passende Netzmaske angegeben werden, damit Netzmaske ind Broadcast für das Interface richtig und nicht class-full gesetzt werden.

Angenommen das Interface `eri0` soll die IP 192.168.2.12 bekommen und das Netz hat eine 27-Bit Maske. In die Datei `/etc/hostname.eri0` enhält folgenden Eintrag:

```
192.168.2.12
```

bzw.

```
eri0
```

falls der Hostname `eri0` z.B. über einen Eintrag

```
192.168.2.12  eri0
```

in der `/etc/hosts` aufgelöst werden kann. Zusätzlich muss die Netzmaske des Netzes noch in der `/etc/netmasks` angegeben werden:

```
192.168.2.0  255.255.255.224
```

Damit wird das Interface nach dem booten richtig initialisiert. Duplex-Verfahren und Übertragungsgeschwindigkeit werden mittels Autonegotiation ausgehandelt. Da das in der Praxis aber oft nicht so klappt wie gedacht und letztendlich das eine Interface auf 100 full und das andere auf 100 halb oder noch grausameren Einstellungen endet, ist es sicherer das Interface fest einzustellen. Dazu müssen mit `ndd` die Treibereinstellungen des Interfaces geändert werden. Ein kleines Initskript welches beim booten ausgeführt wird kann dies erledigen:

```
#!/bin/sh
ndd -set /dev/ce instance 0
ndd -set /dev/ce adv_1000fdx_cap 0
ndd -set /dev/ce adv_1000hdx_cap 0
ndd -set /dev/ce adv_100fdx_cap 1
ndd -set /dev/ce adv_100hdx_cap 0
ndd -set /dev/ce adv_10fdx_cap 0
ndd -set /dev/ce adv_10hdx_cap 0
ndd -set /dev/ce adv_autoneg_cap 0
```

Der erste Befehl stetzt die Instanz fuer `/dev/ce` auf `0`, also alle nachfolgenden `ndd -set` Anweisungen wirken sich auf das Interface `ce0` aus. Der Vollständigkeit halber sei erwähnt, dass für den Fall das alle Instanzen eines Interfaces gleich eingestellt werden sollen, die entsprechenden Anweisungen auch in die `/etc/system` bzw. in die *`interface`*`.cfg` unter `/plattform/$(uname -m)/kernel/drv` eingetragen werden können. Der Weg über ein Initskript ist allerdings der gebräuchlichste und sollte daher auch gewählt werden. Mit `ndd -get` können die aktuellen Einstellungen ausgelesen werden. Einfacher geht das allerdings mit dem Skript `nicstatus.sh`[16]:

```
# nicstatus.sh

      Link:  Auto-Neg:   Status:   Speed:    Mode:  Ethernet Address:
---------------------------------------------------------------------
      eri0         ON        UP    100MB      FDX    0:3:ba:24:6c:a9
```

Die Defaultroute wird über den entsprechenden Eintrag in `/etc/defaultrouter` konfiguriert.

1.6.2 Interfacekonfiguration unter Linux

Unter RHEL spielt sich die Netzwerkkonfiguration in /etc/sysconfig und dessen Unterverzeichnissen ab. Die Grundlage der Netwerkkonfiguration bildet die Datei `/etc/sysconfig/network`. Sie enthält maximal sechs Angaben, in den meisten Fällen aber mindestens diese drei:

[16] http://www.sun.com/bigadmin/scripts/submittedScripts/nicstatus.txt

```
NETWORKING=yes
HOSTNAME=server1.example.com
GATEWAY=192.168.2.1
```

`HOSTNAME` kann auch über DHCP vergeben werden. Ausserdem können noch `NETWORKING_IPV6`, `NISDOMAIN` und `GATEWAYDEV` (Interface über das `GATEWAY` erreicht werden kann) definiert werden falls nötig.

Über die Skripte in `/etc/sysconfig/network-scripts` werden alle nötigen Netzwerinformationen bezogen und gesteuert. Für uns sind an dieser Stelle in erster Linie die `ifcfg-`*`interface`* Skripte interessant.

Um beispielsweise dem Interface `eth0` die IP 192.168.2.13 im Netz 192.168.2.0/27 zu vergeben würde folgende `ifcfg.eth0` ausreichen:

```
DEVICE=eth0
BOOTPROTO=static
IPADDR=192.168.2.12
NETMASK=255.255.255.224
ONBOOT=yes
TYPE=Ethernet
```

Zusätzlich können noch weitere Optionen wie die Broadcast-Adresse, DNS-Server oder Interfacebonding übergeben werden. Interessant ist im Vergleich zur Netzwerkkonfiguration unter Solaris noch die Option `ETHTOOL_OPTS`. Unter Linux werden Duplex-Modus und Übertragungsgeschwindigkeit mit `ethtool` bzw. `mii-tool` gesetzt und abgefragt:

```
# ethtool eth0
Settings for eth0:
        Supported ports: [ MII ]
        Supported link modes:   10baseT/Half 10baseT/Full
                                100baseT/Half 100baseT/Full
                                1000baseT/Half 1000baseT/Full
        Supports auto-negotiation: Yes
        Advertised link modes:  10baseT/Half 10baseT/Full
                                100baseT/Half 100baseT/Full
                                1000baseT/Half 1000baseT/Full
        Advertised auto-negotiation: Yes
        Speed: 100Mb/s
        Duplex: Full
        Port: Twisted Pair
        PHYAD: 1
        Transceiver: internal
        Auto-negotiation: on
        Supports Wake-on: g
        Wake-on: d
        Current message level: 0x000000ff (255)
        Link detected: yes

# mii-tool
eth0: negotiated 100baseTx-FD, link ok
eth1: no link
```

Der Befehl um das Interface `eth0` mit `ethtool` fest auf 100 Mbit full duplex einzustellen lautet `ethtool -s eth0 speed 100 duplex full autoneg off`. Mit `mii-tool` erreicht man dasselbe durch ein `mii-tool -F 100baseTx-FD`.

Um diese Änderung bootfest zu machen könnte man wie unter Solaris die entsprechenden Befehle in ein Initscript schreiben. Einfacher geht es aber über die `ifcfg.`*`interface`* und die Option `ETHTOO_OPTS`:

```
DEVICE=eth0
BOOTPROTO=static
IPADDR=192.168.2.12
NETMASK=255.255.255.224
ONBOOT=yes
TYPE=Ethernet
ETHTOOL_OPTS="autoneg off speed 100 duplex full"
```

1.7 Statische Routen

Oft ist notwendig einzelne Hosts bzw. Netze unabhängig vom Defaultgateway zu routen. Es gibt verschiedene Möglichkeiten um statische Routen einzupflegen, die auch nach einem Reboot der Maschine bestehen bleiben. Letztendlich nutzen wir unter Solaris und Linux den gleichen Ansatz über ein Init-Script.

1.7.1 statische Routen unter Solaris

Der einfachste Weg ohne einen zusätzlichen Routingprozess starten zu müssen führt über ein Init-Script. Man legt dazu z.B. unter `/etc/rc2.d` eine für den Superuser `root` ausführbare Datei `S77routen` an. In diese schreibt man die `route` Befehle die man letztendlich auch auf der Shell verwenden würde.

```
# cat S77routen
#!/sbin/sh
#
# static routes
#
route add net 172.16.0.0 -netmask 255.255.0.0 192.168.230.2 1
route add net 172.19.0.0 -netmask 255.255.0.0 192.168.230.2 1
route add net 192.168.3.0  -netmask 255.255.255.0 192.168.100.51 1
route add net 192.168.222.0 -netmask 255.255.255.0 192.168.220.1 1
route add net 10.17.0.0 -netmask 255.255.0.0 192.168.100.254 1
```

1.7.2 statische Routen unter Linux

Statische Routen in Linux[17] können pro Interface gesetzt werden. Für jedes Interface über das statische Routen gesetzt werden, wird unter `/etc/sysconfig/network-scripts`

[17] Bitte dran denken, dass wir hier von Red Hat Enterprise Linux reden. Bei anderen Distributionen können durchaus andere Vorgehensweisen nötig sein.

eine Datei `route-<interface>` angelegt. Dort werden über die Einträge `GATEWAYn`, `NETMASKn` und `ADDRESSn` die statischen Roten gesetzt, wobei `n` eine Zahl ist und dazu dient die Einträge zu verknüpfen. Sehen wir uns das Ganze mal im Beispiel an. So sieht die aktuelle Routingtabelle aus:

```
# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.2.28    *               255.255.255.240 U     0      0        0 eth0
169.254.0.0     *               255.255.0.0     U     0      0        0 eth0
default         gateway         0.0.0.0         UG    0      0        0 eth0
```

Jetzt füllen wir die Datei `/etc/sysconfig/network-scripts/route-eth0` mit folgendem Inhalt:

```
GATEWAY0=192.168.2.42
NETMASK0=255.255.255.255
ADDRESS0=172.16.10.8

GATEWAY1=192.168.2.23
NETMASK1=255.255.255.255
ADDRESS1=10.10.10.40
```

Nach einem Reboot bzw. Restart des Netzwerks mittels `service network restart` sieht die Routingtabelle wie folgt aus:

```
# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
172.16.10.8     192.168.2.42    255.255.255.255 UGH   0      0        0 eth0
10.10.10.40     192.168.2.23    255.255.255.255 UGH   0      0        0 eth0
192.168.2.28    *               255.255.255.240 U     0      0        0 eth0
169.254.0.0     *               255.255.0.0     U     0      0        0 eth0
default         gateway         0.0.0.0         UG    0      0        0 eth0
```

Die beiden neuen Hostrouten sind jetzt aktiv und bootfest.

Stattdessen kann man natürlich auch denselben Weg wie ich ihn oben für Solaris beschrieben habe gehen, und unter `/etc/rc3.d` ein entsprechendes Init-Script anlegen[18].

```
#!/bin/sh
#
# static routes
#
route add -net 172.16.0.0 netmask 255.255.0.0 gw 192.168.230.2 metric 1
route add -net 172.19.0.0 netmask 255.255.0.0 gw 192.168.230.2 metric 1
route add -host 192.168.3.15 gw 192.168.100.51 metric 1
```

Alternativ können die Routen auch in die `/etc/rc.local` gesetzt werden.

[18] Oder, wenn man genau ist, ein Script in `/etc/init.d/` anlegen und in `/etc/rc3.d` einen Symlink darauf setzen.