

Linux im Netzwerk - ein "Mini"-Tutorial (Teil 2)

[Jana Jaeger](#)

[&content/server/fire1.html">Teil 1](#) hat sich mit den nötigsten Grundlagen zu Kommunikation unter Linux im Netz befaßt. Im zweiten Teil des Mini-Tutorials wird es darum gehen, wie ein einfaches Beispielnetz mit Internetzugang aufgebaut sein kann und welche wichtigen Konfigurationsarbeiten vorzunehmen sind.

Inhaltsverzeichnis

- [Ein Netzwerk aufbauen - Was ist zu tun?](#)
- [Konfigurationsarbeiten](#)
- [Masquerading einrichten](#)
- [Konfigurationsdateien und Startup-Skripte](#)
- [Routing unter SuSE Linux](#)
- [Ausblick](#)
- [Literatur](#)

Ein Netzwerk aufbauen - Was ist zu tun?

Aus drei Arbeitsplatzrechnern und einem vierten Rechner, der als Zugang zum Internet dient, soll ein ganz einfaches Netzwerk aufgebaut werden. Technische Feinessen oder die konkret zu verwendende Hardware bleiben (ersteinmal) außen vor. Bevor es losgeht, müssen folgende Punkte bedacht sein:

Wie sollen die einzelnen Rechner heißen?

Jeder Name darf nur einmal in diesem Netz vorkommen und sollte möglichst kurz und prägnant sein. Die Rechner in unserem Beispiel sollen wallace, gromit, sean und nick heißen.

Wie heißt die Domain, der diese Rechner angehören?

Soll ein Rechner korrekt adressiert werden, muß sein voller Name, die folgenden wichtigen Elemente enthalten: Rechnernamen, Domainname und Top-Level-Domain. Die Domain, der unsere Beispielrechner angehören, soll home heißen, als Top-Level-Domain wird nil gewählt. Das bedeutet konkret, gromit heißt mit vollem Namen: gromit.home.nil. Die Top Level Domain nil ist freie Erfindung, um Verwechslungen mit real existierenden Netzen auszuschließen. Domainnamen fallen nicht einfach vom Himmel, im einfachsten und normalen Fall regeln sich solche Dinge über den Provider (ISP).

Wie lauten die IP-Adressen der Rechner?

Für jeden Rechner muß mindestens eine IP-Adresse angegeben werden, die ihn eindeutig charakterisiert. Für nick werden sogar zwei IP-Adressen angegeben, da er zwei Netzwerkkarten und damit Schnittstellen zum lokalen Netz wie auch zum Internet hat.

nick 192.168.17.1 / 132.147.11.4

wallace	192.168.17.2
gromit	192.168.17.3
sean	192.168.17.4

Sogar im internen Netz greifen einige Regeln des großen Internets, die RFCs (Request For Comments), die dafür sorgen, daß gewisse Standards im großen, ständig wachsenden Netz gewahrt bleiben und das Internet nicht im Chaos versinkt. Für die Vergabe interner (privater) IP-Adressen ist dies RFC 1918 (Address Allocation for Private Internets). Damit Rechner miteinander kommunizieren können, müssen sie unterschiedliche IP-Adressen haben, wir können uns also nicht einfach die IP von `www.suse.de` (213.95.15.200) schnappen, da dann keine Kommunikation zwischen unserem Rechner und `www.suse.de` möglich ist. RFC 1918 beschreibt eine Reihe von Adressen, die nirgendwo im Internet benutzt werden, und deswegen explizit für privaten Gebrauch zur Verfügung stehen. Natürlich können wir mit diesen Adressen dementsprechend auch nicht ins Internet. Unser Router muß deswegen die privaten Adressen durch eine offizielle Adresse ersetzen, das Ganze nennt sich NAT (Network Address Translation) und wird in einem späteren Abschnitt genauer erklärt. Die privaten Adressbereiche sind nun:

- 10.0.0.0 - 10.255.255.255 (10/8 Präfix), Class A
- 172.16.0.0 - 172.31.255.255 (172.16/12 Präfix), Class B
- 192.168.0.0 - 192.168.255.255 (192.168/16 Präfix), Class C

Für unser privates Netz haben wir uns daraus 192.168.17.0/24 ausgewählt. 192.168er Netzwerkadressen sind grundsätzlich für kleinere Netze geeignet. Zu den Unterschieden zwischen Class A-, B-, C-Netzen sei auf [&content/server/fire1.html">Teil 1](#) verwiesen.

Um `nick` per statischem Routing mit dem Internet zu verbinden, sind noch einige wichtige Angaben über das Nachbarnetz notwendig:

- IP und Name des nächsten Gateways: `132.147.11.8` und `ginger`

Wie heißt der Gateway?

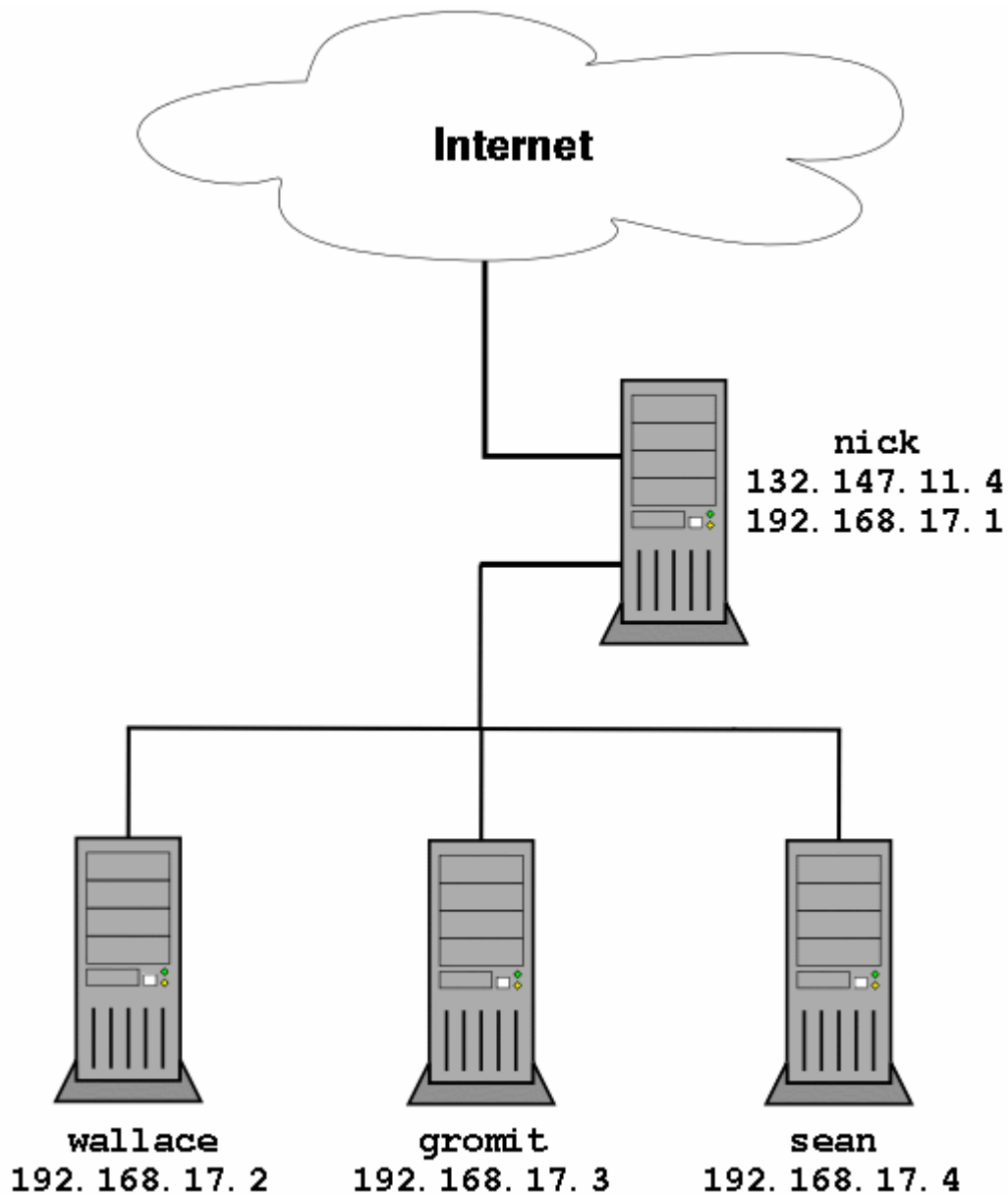
Wenn sich, wie in unserem Beispiel, ein Rechner im lokalen Netz befindet, der zusätzlich mit einem anderen Netz verbunden ist, muß er auch für dieses Netz über Namen und IP-Adresse verfügen. Im Beispiel ist dies `nick`.

Welche Netzwerkmaske?

Für die Zuordnung einzelner Netzwerkadressen zu bestimmten Netzen und Subnetzen braucht man Netzmasken (siehe [&content/server/fire1.html">Teil 1](#) des Tutorials). Die Netzmaske für unser Netz soll `255.255.255.0` (192.168.17.0/24) sein.

Befindet sich im Netz ein Nameserver? Wo?

Die Verwandlung von Rechnernamen in IP-Adressen wird vom *Domain Name Service* (DNS) vorgenommen. Der entsprechende Server kann entweder über das Internet erreichbar sein oder im lokalen Netz liegen. Der Einfachheit halber soll im Beispiel der Nameserver auf `nick` laufen.



Konfigurationsarbeiten

Nach diesen Vorüberlegungen geht es jetzt zur Sache. Die folgenden Schritte beziehen sich auf die Netzwerkkonfigurationsarbeiten auf das Vorgehen unter SuSE Linux 7.1 mittels *YaST2*. Alle Schritte betreffen ausschließlich den oben skizzierten Aufbau **ohne** ISDN und Modemkonfiguration.

Die Beispielkonfiguration könnte so wie beschrieben für jeden der Rechner im Netz gelten. Einstellungen, die speziell für den Gateway vorzunehmen sind, werden in **Fettschrift** abgesetzt und sollten erst nach Abschluß der normalen Konfiguration vorgenommen werden. Erst diese Konfigurationsschritte machen aus **nick** den eigentlichen Gateway. Über das *YaST2* Kontrollzentrum erreichen Sie das Modul Netzwerkkarte einrichten. Anhand der Screenshots wird verdeutlicht, welche Einstellungen wo gesetzt werden müssen und welche Konfigurationsdateien von diesen Änderungen jeweils betroffen sind.



Der erste Dialog öffnet sich mit einer Übersicht über die bereits im System vorhandenen und (bei der Installation erkannten) Netzwerkkarte. Da der Rechner jetzt mit seiner Netzwerkkarte mit den restlichen Maschinen bekanntgemacht werden soll, sollten Sie den Button Bearbeiten drücken. Der folgende Dialog verlangt von Ihnen eine Angabe der IP-Adresse [siehe Schritt 5](#).

[/etc/rc.conf](#)
[ig](#)

nicks als der Gateway des Netzes soll über eine weitere Netzwerkkarte mit dem Internet in Verbindung stehen, deshalb muß diese erstmal in die Konfigurationsdatei samt aller Angaben wie Gerätenamen (in unserem Fall `eth1` für die zweite Netzwerkkarte) und IP-Adresse aufgenommen werden. Drücken Sie Hinzufügen.

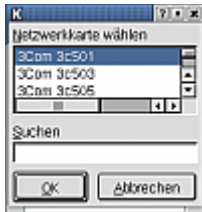


In diesem und den drei folgenden Schritten wird **nicks** zweite Netzwerkkarte korrekt konfiguriert. Für "Netzwerkcontroller" werden Sie die Anfrage "Nicht erkannte Karte verwenden" und "Karte einrichten" gestellt bekommen. Über den Button "Karte einrichten ..." gelangen Sie in den nächsten Dialog.



Der Dialog "Manuelle Netzwerkkonfiguration" benötigt Ihre Angaben zu Netzwerktyp (der eingestellte Default ist hier "Ethernet") und Gerätenummer (für nicks zweite Netzwerkkarte wird, sofern Sie nichts anderes anwählen, "1" eingetragen). Weiterhin müssen das entsprechende Kernelmodul und seine Optionen hier eingetragen werden. Wählen Sie den Button "Treffen Sie eine Auswahl aus der Liste" und suchen Sie aus der Auswahl Ihre Netzwerkkarte aus. Ist sie in der Liste enthalten, werden Modul und Optionen nach Bestätigung der Auswahl automatisch in die entsprechenden Felder übernommen. Mit "Weiter" gelangen Sie in den nächsten Dialog.

[/etc/rc.conf](#)
[ig](#)



Auswahlliste unterstützter Netzwerkkarten ...



Die erste Netzwerkkarte (internes Netz) bekommt in diesem Schritt eine IP-Adresse und ihre Subnetzmaske zugewiesen. Wenn sich im Netz, wie in diesem einfachen Fall, kein DHCP-Server zur dynamischen Adressvergabe befindet, muß die Option "Konfiguration der statischen Adresse" ausgewählt werden. Nun tragen Sie `nicks` interne Netzwerkadresse `192.168.17.1` und die vereinbarte Subnetzmaske `255.255.255.0` ein.

Das Vorgehen zur Konfiguration der nach außen gewandten IP-Adresse erfolgt analog. Entweder Ihr Provider hat Ihnen mitgeteilt, welche Einstellungen Sie für eine statische Konfiguration eintragen müssen oder aber er hat einen DHCP Server und Sie brauchen sich um die IP-Adressvergabe nicht weiter zu kümmern.

In beiden Fällen folgen jetzt die gleichen Arbeitsschritte. Zuerst werden Hostname und Nameserver und anschließend das Routing konfiguriert.

[/etc/hosts](#)



Die entsprechenden Masken erwarten jetzt eine Eingabe des Hostnamens (hier `nick`) und der Domain (`home.nil`). Diese Angaben sind obligat, da sie die eigentliche "Identität" der Maschine festlegen. Ist ein Nameserver im Netz (in diesem Fall auf `nick` selber) wird dessen IP-Adresse (`192.168.17.1`) und die zu durchsuchende Domain `home.nil` eingefügt.

Für `nicks` zweite Netzwerkkarte werden keine extra Konfigurationsarbeiten notwendig. Es werden automatisch die systemweiten Einstellungen (s.o.) übernommen.

[/etc/resolv.conf](#)



Jetzt fehlt nicht mehr viel zum vernetzten Rechner. Die Angabe des Standardgateways und der Routing Tabelle fallen im Falle der nach innen orientierten Netzwerkkarte von `nick` flach, d.h. hier werden keine Einträge gemacht. Alle anderen Rechner im Netz bekommen hier

[/etc/route.conf](#)

	<p>mindestens die Angabe von <code>nick</code> (IP 192.168.17.1) mit eingetragen.</p> <p>Der Gateway <code>nick</code> ist seinerseits wiederum Teil eines größeren Netzwerks, deshalb wird hier bei Standardgateway der Gateway des Providers <code>park.nil</code>, 132.147.11.8 (<code>ginger</code>), eingetragen. Optional k "Konfiguration für Experten" vorgenommen werden.</p>	
--	---	--

Um `nick` endgültig in die Schnittstelle zwischen lokalem Netz und Internet zu verwandeln, muß mindestens noch ein wichtiger Eintrag in der `/etc/rc.config` (beispielsweise über den `rc.config` Editor von *YaST2*) vorgenommen werden. Nach diesem Schritt ist `nick` so konfiguriert, daß er Pakete entsprechend seiner Routing Tabelle von außen nach innen und in umgekehrter Richtung weiterleiten kann. Alle anderen Maschinen im Netz funktionieren nicht als Router und brauchen dementsprechend auch diese Einstellung nicht.

```
#
# runtime-configurable parameter: forward IP packets.
# Is this host a router? (yes/no)
#
IP_FORWARD="yes"
```

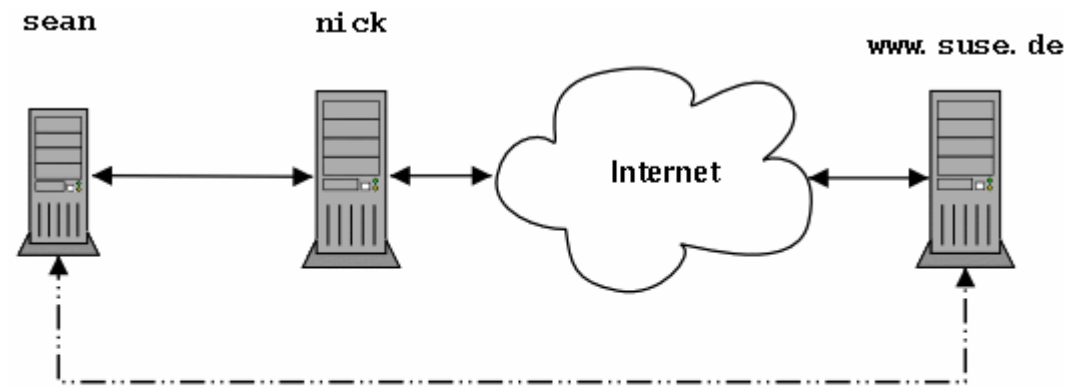
Nach diesem Schritt können Sie *YaST2* verlassen.

Achtung: An dieser Stelle könnte auch der Nameserver auf `nick` eingerichtet werden, den das Netz zur reibungslosen Kommunikation mit der restlichen Welt dringend benötigt. Da sich diese Thematik aber nicht einfach in ein paar Zeilen abhandeln läßt, sei auf einen der nächsten Artikel in dieser Serie verwiesen, in dem die Einrichtung eines lokalen Domain Servers mit Caching beschrieben wird.

Masquerading einrichten

Zum Schluß noch eine Einstellung, die `nick` als das "Tor" zum Internet betrifft: die Umsetzung lokaler IP-Adressen in internetweit gültige Adressen, also NAT bzw. Masquerading.

Das Prinzip dahinter ist eigentlich recht einfach: Einzig unser Gateway `nick` hat eine internetweit eindeutige IP-Adresse und kann somit direkt mit jedem beliebigen Rechner im Internet kommunizieren, da seine Adresse eindeutig ist. Unsere drei internen Rechner haben diese Fähigkeit noch nicht, sie besitzen IPs, die nur innerhalb vom lokalen Netz `home.nil` eindeutig sind, und innerhalb beliebig vieler anderer privater Netzwerke auch so vergeben sein können. Damit also auch diese Rechner einen Zugang zum Internet bekommen, muß ihre interne Adresse von `nick` bei ausgehenden Datenpaketen in eine eindeutige IP-Adresse umgewandelt werden und bei eingehenden Paketen eine entsprechende Verteilung auf die internen Empfängeradressen gewährleistet werden.



```

Host: sean          Host: sean -> nick
Quellport: QP(sean) Quellport: QP(sean) -> QP(nick)
Host: www.suse.de  Host: www.suse.de
Zielport: 80       Zielport: 80

```

Die obige Abbildung soll das Vorgehen ein wenig klarer machen. Angenommen, sean sendet eine HTTP-Anfrage an `www.suse.de` (gestrichelte Linie). Die von sean gesendete Anfrage wird über nick ins Internet weitergeleitet (durchgezogene Linie). Hierbei ersetzt nick im Header des ausgehenden Pakets die Absenderadresse von sean durch seine eigene Adresse und ändert auch den Quellport des Pakets ab. Von nun an trägt das Paket im Internet eine voll gültige IP-Adresse im Header und kann problemlos an Port 80 von `www.suse.de` zugestellt werden. Das Antwortpaket an sean muß wiederum nick passieren. Die Zieladresse weist dieses Paket als an nick gerichtet aus - wie wird jetzt sichergestellt, daß sean jemals die Antwort auf seine Anfrage erhält? Die Antwort liegt in der bei Ausgang der Anfrage geänderten Quellportnummer. nick kann diese Portnummer mithilfe entsprechender interner Tabellen als Aufforderung deuten, Pakete mit dieser Portnummer im Header an sean weiterzureichen.

Ab SuSE Linux 7.1 ist die beste Lösung für einfaches Masquerading die Benutzung des Pakets *personal-firewall*. Es läßt sich sehr einfach konfigurieren, bietet aber gleichzeitig einen gewissen Schutz für das interne Netz, da am äußeren Device alle eingehenden Verbindungen abgelehnt werden. *personal-firewall* greift zwar auf die *ipchains* Funktionen des 2.2er Kernels zurück, kann aber auch unter Kernel 2.4.x verwendet werden, wenn das Kernelmodul *ipchains* geladen wird. Die konzeptionellen Unterschiede zwischen *ipchains* und *iptables* hier an dieser Stelle zu diskutieren, würde den Rahmen dieses Artikels sprengen. Deshalb an dieser Stelle nur die nötigsten Angaben zur Konfiguration von *personal-firewall* - der Rest folgt später ;-) Ebenso gibt es zum Thema Paketfilter auch noch eine Menge mehr zu sagen...

Um diese Lösung zu benutzen, muß das Paket `personal-firewall` aus der Serie `sec` installiert sein. Es gehört zum Umfang der Standardinstallation.

Zur Konfiguration muß nur eine Variable angepasst werden:

```
REJECT_ALL_INCOMING_CONNECTIONS in der Datei
/etc/rc.config.d/security.rc.config.
```

Mögliche Werte sind:

`no`

Alle Regeln werden gelöscht, Filterung und Masquerading finden nicht statt.

`yes`

Alle eingehenden Verbindungen werden abgelehnt.

`iface`

Das Device (die Devices) an welchen eingehende Verbindungen abgelehnt werden sollen. z.B.

ipp0 wenn Sie ISDN benutzen.

masq

wenn der Verkehr des internen Netzes maskiert (NAT) werden soll. Damit wird alles maskiert, was über ein Interface kommt, welches nicht geblockt wird. Das heißt bei `REJECT_ALL_INCOMING_CONNECTIONS="yes"` macht masq keinen Sinn. Außerdem muß die Option `IP_FORWARD` in `etc/rc.config` auf "yes" gesetzt sein.

Das heißt in unserem Fall müßte der Eintrag folgendermaßen aussehen:

```
REJECT_ALL_INCOMING_CONNECTIONS="eth1 masq"
```

Somit werden alle Pakete, die über `nick` weitergeleitet werden, maskiert und alle eingehenden Verbindungen zu `eth1` werden abgelehnt.

Ready, steady, go ...!

Das in groben Zügen fertig konfigurierte Netzwerk - natürlich müssen auch `wallace`, `gromit` und `sean` - entsprechend dem oben beschriebenen Muster noch konfiguriert werden - wird jetzt mit der Eingabe von

```
rcnetwork start
```

gestartet. Nun sollte unser kleines Netzwerk vollen Zugang zum Internet genießen und alle Teilnehmer miteinander kommunizieren können.

Konfigurationsdateien und Startup-Skripte

Die grundlegendsten Netzwerkeinstellungen sind jetzt mit YaST vorgenommen. Die wichtigsten Konfigurationsdateien hier kurz im Überblick:

`/etc/rc.config` (SuSE-spezifisch)

Die zentrale Konfigurationsdatei von SuSE Linux. Die meisten Einstellungen der Netzwerkkonfiguration fließen hier mit ein. Sollen später noch bestimmte Funktionen hinzugeschaltet werden, müssen diese nachträglich von Hand aktiviert werden. Seit SuSE 7.1 erreicht man dies über den `rc.config`-Editor im *YaST2*-Kontrollcenter.

Die wichtigsten (schon automatisch per *YaST2* erledigten) Einträge sind hier am Beispiel eines der normalen Hosts (z.B. `gromit`) beschrieben:

```
#
# IP Adresses
#
IPADDR_0="192.168.17.3"
IPADDR_1=""
IPADDR_2=""
IPADDR_3=""

#
# network device names (e.g. "eth0")
#
NETDEV_0="eth0"
NETDEV_1=""
NETDEV_2=""
NETDEV_3=""

#
```

```

# parameteres for ifconfig, simply enter "bootp" or "dhcpclient" to use the
# respective service for configuration
# sample entry for ethernet:
# IFCONFIG_0="192.168.81.38 broadcast 192.168.81.63 netmask 255.255.255.224"
#
IFCONFIG_0="192.168.17.3 broadcast 192.168.17.255 netmask 255.255.255.0"
IFCONFIG_1=""
IFCONFIG_2=""
IFCONFIG_3=""

#
# runtime-configurable parameter: forward IP packets.
# Is this host a router? (yes/no)
#
IP_FORWARD="no"

#
# SuSEconfig can do some checks and modifications in /etc/hosts.
# If this is not wanted, set the following variable to 'no' (yes, no).
#
CHECK_ETC_HOSTS="yes"

#
# If CHECK_ETC_HOSTS is set to yes, SuSEconfig sorts your
# /etc/hosts. But in some cases this may be unwanted. So here is a
# flag, where you can configure if /etc/hosts should be "beautified".
# (yes/no)
#
BEAUTIFY_ETC_HOSTS="yes"

# The fully qualfied hostname of this computer
#
FQHOSTNAME="gromit.home.nil"

#
# Space sep. list of nameservers that should be used for /etc/resolv.conf
#
NAMESERVER="192.168.17.1"

#
# Number of network cards: "_0" for one, "_0 _1 _2 _3" for four cards
#
NETCONFIG="_0"

#
# Domain searchlist that should be used in /etc/resolv.conf
#
SEARCHLIST="home.nil"

```

Die beiden Variablen NAMESERVER und SEARCHLIST werden per *YaST2* automatisch eingetragen. Allerdings können sie auch alternativ, wie unten im Auszug der `/etc/resolv.conf` manuell gesetzt werden.

`/etc/hosts`

Hier werden Rechnernamen den entsprechenden IP-Adressen zugeordnet. Wird kein Nameserver verwendet, so müssen in dieser Datei alle Rechner vermerkt sein, mit denen IP-Verbindungen aufgebaut werden sollen. Die Syntax sieht wie in der gezeigten Ausgabe der `/etc/hosts` aus.

```

#
# hosts          This file describes a number of hostname-to-address
#               mappings for the TCP/IP subsystem.  It is mostly
#               used at boot time, when no name servers are running.
#               On small systems, this file can be used instead of a
#               "named" name server.

```

```
# Syntax:
#
# IP-Address  Full-Qualified-Hostname  Short-Hostname
#

127.0.0.1      localhost
192.168.17.1   nick.home.nil    nick
192.168.17.2   wallace.home.nil wallace
192.168.17.3   gromit.home.nil  gromit
192.168.17.4   sean.home.nil    sean
```

/etc/resolv.conf

Sollen die Dienste des DNS verwendet werden, werden einige wichtige Funktionen der glibc verwendet, die in der *resolver* Bibliothek zusammengefaßt sind. Damit diese Routinen die Namensauflösung korrekt vornehmen können, muß ihnen mitgegeben werden, in welcher Domain der Rechner selber steht (Schlüsselwort **search**) und wie die Adresse des Nameservers (Schlüsselwort **nameserver**) ist. Es ist möglich, hier mehrere zu durchsuchende Domains anzugeben oder auch mehrere Nameserver einzutragen. Kommentare werden, wie üblich mit # auskommentiert.

Der typische Auszug einer /etc/resolv.conf sieht so aus:

```
#
# /etc/resolv.conf
#
search home.nil
nameserver 192.168.17.1
```

Diese Einstellungen werden per *YaST2* eingetragen.

Alle für unser kleines Beispielnetzwerk relevanten Startupskripte werden kurz in ihrer Funktion vorgestellt:

/etc/init.d/network

Mit diesem Skript wird bei Start des Systems die Konfiguration der Netzwerk- Hard- und Software vorgenommen. Unter anderem greift dieses Skript auch auf die mit *YaST2* in die /etc/rc.config gemachten Einträge zu und wertet sie aus.

/etc/init.d/route

Dieses Skript dient dem statischen Routen im Netzwerk. Hierfür wertet es die Konfigurationsdatei /etc/route.conf aus.

Routing unter SuSE Linux

In Zusammenhang mit dem Beispielnetz soll aus Gründen der Einfachheit nur auf statisches Routing eingegangen werden. "Statisch" bedeutet, daß alle möglichen Routen per Routing Tabellen an den einzelnen Netzknotenpunkten festgelegt sind. Für die Zwecke unseres "home" Netzes sind nur ein paar Einstellungen wichtig:

- a. Route zu einem anderen Rechner
- b. Route zu einem anderen Rechner über ein Gateway
- c. Route zu einem Netzwerk

Die Ansicht einer `/etc/route.conf` Datei ähnelt in wesentlichen Zügen der Ausgabe des Befehls `/sbin/route`, die die aktuelle Routing Tabelle des Kernels liefert.

```
tux@sean:~ > /sbin/route
Kernel IP routing table
Destination Gateway          Genmask      Flags Metric Ref    Use Iface
192.168.17.0 *                255.255.0.0 U        0     0     0 eth0
loopback      *                255.0.0.0   U        0     0     0 lo
default       nick.home.nil    0.0.0.0     UG       0     0     0 eth0
```

Kurz und gut besagt diese Routing Tabelle, daß alle Pakete an Teilnehmer im internen Netz direkt über die Netzwerkkarte (`eth0`) verschickt werden, ohne über den Gateway verteilt werden zu müssen. Die `loopback` Zeile steht für das Loopback Device - einer Art definierter geschwindelter Netzwerkkarte für Programme, die zwar Netzfunktionen nutzen, aber trotzdem nie über den lokalen Rechner hinausgehen. Solche Programme funktionieren auch auf nicht vernetzten Rechnern. Die letzte Zeile besagt, daß alles andere, was nicht im internen Netz zugestellt werden soll, über `nick` an die Außenwelt gelangt. Abzüglich der Angaben `Flags`, `Metric`, `Ref`, `Use` ist der Aufbau der `/etc/route.conf` gleich. Die Einträge gliedern sich so:

- alle Zeilen, die entweder leer sind oder mit einem `#` beginnen, werden komplett ignoriert. Ein "richtiger" Eintrag enthält mindestens zwei und maximal vier Spalten.
- Die erste Spalte enthält das Ziel der Route ("Destination"). Das kann eine IP-Adresse oder auch ein voll qualifizierter Rechner- bzw. Netzname sein. Die letztere Option funktioniert nur bei erreichbaren Namenservern. `default` steht für den Default-Gateway, bitte hier immer `default` und keineswegs `0.0.0.0` angeben.
- In der zweiten Spalte steht immer die IP-Adresse, ein Platzhalter (`0.0.0.0`) oder der volle Rechnername eines Gateways.
- In der dritten Spalte wird die Netzwerkmaske für Netzwerke oder Rechner hinter einem Gateway angegeben. Für einen Rechner hinter einem Gateway sollte die Angabe `255.255.255.255` gemacht werden, damit genau dieser Rechner mit seiner einzigartigen IP-Adresse die "Post" weitergeleitet bekommt.
- In der letzten Spalte werden die am lokalen Rechner vorhandenen Netzwerkgeräte eingetragen. Also in unserem Beispiel nur `lo` für das Loopbackdevice und `eth0` bzw. für `nick` auch noch `eth1` eintragen.

So könnte `nicks route.conf` aussehen:

```
# Destination      Dummy/Gateway      Netmask            Device
#
# Net Devices
# Not needed with Kernel 2.2 or newer:
# 127.0.0.0         0 0 0 0           255.0.0.0         lo
# 192.168.17.0     0.0.0.0           255.255.255.0    eth0
# 132.147.11.0    0.0.0.0           255.255.255.0    eth1
#
# Gateway
#
default           132.147.11.8
```

Damit sind alle Einträge in die relevanten Konfigurationsdateien gesetzt. Wenn sich bei der Konfiguration oder dem physikalischen Aufbau keine Fehler eingeschlichen haben, müßte das Netz jetzt betriebsbereit sein. Jeder Rechner im Beispielnetz kann mit seinen unmittelbaren Nachbarn kommunizieren und über den Gateway auch Kontakt zu anderen Netzen aufnehmen.

Ausblick

Was kann passiert sein, wenn trotz (vermeintlich) umsichtiger Konfiguration nicht alles im Netz so


klappt, wie es sollte? Wie kann man Fehlern auf die Spur kommen? Welche Werkzeuge stehen dem Netzwerker zur Verfügung?

Wie richte ich DNS ein?

Mit diesen Fragen beschäftigt sich Teil 3 und 4 des Netzwerk-Tutorials.

Steht das Netzwerk ersteinmal soweit, gibt es ein paar Werkzeuge, mit denen es (relativ) einfach, aber wirkungsvoll gegen Angriffe abgesichert werden kann. Wie? Das wird Teil 5 des Tutorials behandeln. Bis dann ;-)

Literatur

- [Linux Network Administrator's Guide](#) von Olaf Kirch, die erste Ausgabe (von 1996) gibt es auch in einer deutschen Version als [Linux - Wegweiser für Netzwerker](#)
- Grundlegendes zum Thema Linux im Netzwerk findet sich auch im Handbuch (Installation, Netzwerk und Know-How) von SuSE Linux 7.1 (Professional), Kapitel 5.
- Die RFCs können im Verzeichnis `/usr/share/doc/rfc/` nachgeschlagen werden, sofern das Paket `rfc` aus der Serie `doc` installiert ist. Alternativ sind die Standards über das Internet unter <http://www.rfc-editor.org/index.html> abrufbar. 

Linux auf dem Server 24.04.2001