# Versuche im Netzwerk auf Schicht 2 und 3

Peter Maass, Michael Dienert

6. April 2011

## Inhaltsverzeichnis

**1 Ein Testnetz mit 2 Routern** **1**
1.1 Verkabelung des Testnetzes . . . . . 1
1.2 Routing mit dem Linux-Kernel . . . . . 1
1.3 Ausgestorbene, afrikanische Wildtiere . . . . . 3

**A Eine mögliche Adressaufteilung** **4**

**B Ganz knappe IOS-Anleitung** **5**
B.1 IOS . . . . . 5
B.2 Beispiele für die Grundkonfiguration . . . . . 6

**C Konfigurieren von Interfaces** **7**

# 1 Ein Testnetz mit 2 Routern

Die Abbildung 1 zeigt eine einfache Netztopologie mit insgesamt 3 Subnetzen, die mit 2 Routern verbunden sind.

## 1.1 Verkabelung des Testnetzes

Verkabeln Sie ein Testnetz unter Zuhilfenahme der Experimental-Netzwerkdosen (schwarz beschriftet), der Patchfelder im Datenschrank und der Cisco-Switches. Die beiden Router können direkt am Patchfeld verbunden werden. Dazu benötigt man evtl. ein Cross-Over-Kabel.

## 1.2 Routing mit dem Linux-Kernel

Zuerst wollen wir die Routingfunktion eines Rechners mit Linux-Kernel testen.
Es gilt:

| Ein Router ist ein vollständiger Rechner mit CPU, RAM, ROM und Betriebssystem |
|---|

Umgekehrt gilt damit natürlich auch:

| Jeder normale PC mit Linux-, Windows oder BSD-Kernel kann *ohne* Sondersoftware als Router verwendet werden. |
|---|

Aufgaben:

1. Zuerst sind die drei Netze `A`, `B` und `Verbindungsnetz` zu bestimmen (Netzadresse, Maske, Broadcastadresse). Dabei dürfen die Netze selbstverständlich *nicht* überlappen.

   Die Adressen sollen aus dem Bereich 172.16.0.0/16 entnommen werden.

2. Im nächsten Schritt werden die Interfaces (eth1 und eth2) der beteiligten Rechner mit IP-Adressen konfiguriert.

   Dabei sollen die Routeradressen immer die höchstmöglichen und die Adressen der Arbeitsplatzrechner immer die niedrigsten Adressen in ihrem Subnetz sein.

   Unter Linux konfiguriert man eine Ethernet-Schnittstelle so:

   | **ifconfig** *schnittst.* **netmask** *maske* **broadcast** *bc-Adr.* *ip-Adr.* **up** |
   |---|

   Mit

   | **ifconfig** *schnittstelle* |
   |---|

   kann man sich anschliessend die Konfiguration einer bestimmen Schnittstelle ansehen.

   Um die Konfiguration zu testen, muss folgendes funktionieren:

   (a) innerhalb eines Subnetzes muss ein Rechner mit ping erreichbar sein

   (b) ein Rechner in einem fremden Subnetz darf auf die ICMP-Requests von ping nicht antworten. D.h. er darf sich nicht anpingen lassen.

   Syntax des ping-Kommandos:

   | **ping** *entfernteAdresse* |
   |---|

3. Damit man nun auch die Rechner in den entfernten Subnetzen erreichen kann, muss das Routing aktiviert werden.

   Nach dem Start eines Linux-Kernels ist das Routing ausgeschaltet und muss mit folgendem Kommando aktiviert werden:

   | **echo 1 > /proc/sys/net/ipv4/conf/all/forwarding** |
   |---|

   Bei der Eingabe des langen Pfads sollte man unbedingt die TAB-Taste zu Hilfe nehmen, das spart Zeit und vermeidet Tippfehler.

   Ob das Kommando erfolgreich war, kann man so testen:

| **cat /proc/sys/net/ipv4/conf/all/forwarding** |
|---|

Zur Erklärung: alle Verzeichnisse unterhalb des Verzeichnisses /proc stehen nicht wirklich auf der Festplatte sondern werden nur virtuell dargestellt, als wären sie auf einem Datenträger. In Wirklichkeit bilden die in /proc enthaltenen Datein den Zustand des Kernels ab und stellen so eine bequeme Schnittstelle zum Kernel dar.

4. Der Kernel wäre jetzt bereit zu routen, aber damit er richtig routen kann, müssen Einträge in der Routingtabelle stehen.

   Mit

   | **netstat -rn** |
   |---|

   kann man sich die Kernelroutingtabelle anzeigen lassen.

   Und mit

   | **route add -net** *zielnetz* **netmask** *maske* **gw** *gateway* |
   |---|

   wird eine neue Route eingetragen.

   Mit

   | **route del -net** *zielnetz* **netmask** *maske* **gw** *gateway* |
   |---|

   wird sie wieder gelöscht.

   Tragen Sie also nun die richtigen Routen in die Tabellen von Router R1 und R2 ein. Dabei muss man unbedingt berücksichtigen:

   | Die Route von A nach B beinhaltet **nicht** automatisch die Route von B nach A. |
   |---|

5. Ist alles richtig konfiguriert, sollte man von Host A Host B mit dem Ping-Kommando erreichen.

   Eine genauere Übersicht über den Weg, den die ICMP-Pakete von ping zurücklegen, liefert das Kommando

   | **traceroute** *zielnetz* |
   |---|

## 1.3 Ausgestorbene, afrikanische Wildtiere

Das Routing mit dem Linuxkernel funktioniert so gut, dass ein Entwicklerteam eine Software-Suite geschrieben hat, die die Konfiguration eines Linux-Routers erleichtert und neben dem statischen Routing (eintragen der Routen von Hand) auch Routingprotokolle zur Verfügung stellt, die die Routen dynamisch der Netztopologie anpassen und in die Kerneltabellen eintragen.

Dieses Softwarepaket ist nach der ausgestorbenen Zebra-Art **Quagga** benannt [1].

Augaben:

[1] Quaggas waren wohl keine eigene Art, sondern genetisch den Zebras so ähnlich, dass man sich im Moment an einer Rückzüchtung versucht.

1. Installieren der Quagga-Software. Folgende Schritte sind durchzuführen:

   (a) Das Paketverwaltungssystem aktualisieren:

   ```
   aptitude update
   ```

   (b) Die quagga-Pakete installieren

   ```
   aptitude install quagga quagga-doc
   ```

   (c) Ein paar einleitende Konfigurationsschritte: in der Datei **/etc/quagga/daemons** müssen die Schalter von **zebra** und **ospfd** auf 'yes' gesetzt werden:

   ```
   cd /etc/quagga
   gedit daemons
   touch zebra.conf
   touch ospfd.conf
   export VTYSH_PAGER=more
   /etc/init.d/quagga restart
   vtysh
   ```

   (d) Mit dem Absetzen des Kommandos **vtysh** verlassen Sie die Linux-Kommandozeile und befinden sich fortan in einer Umgebung, die das IOS von Cisco emuliert. Mit diesem IOS-Nachbau kann nun der Router konfiguriert werden. Im Anhang ist eine Kurzeinführung zu IOS und hier:

   http://msv.wara.de/ dienert/iosKommandoRef/ios.pdf

   **Achtung! Wichtiger Unterschied von quagga zu IOS:**
   Bei quagga muss eine Adresse in der CIDR-Schreibweise angegeben werden.
   Z.B.: **quaggaRouter(config-if) ip address 172.16.4.2/22**

# A Eine mögliche Adressaufteilung

| | | |
|---|---|---|
| | Netzadresse: | 172.16.0.0 / 22 |
| | Broadcast: | 172.16.3.255 / 22 |
| **Netz B** | Maske: | 255.255.252.0 |
| | Host: | 172.16.0.1 / 22 |
| | GW: | 172.16.3.254 / 22 |
| | | |
| | Netzadresse: | 172.16.4.0 / 25 |
| | Broadcast: | 172.16.4.127 / 25 |
| **Netz A** | Maske: | 255.255.255.128 |
| | Host: | 172.16.4.1 / 25 |
| | GW: | 172.16.4.254 / 25 |

| | | |
|---|---|---|
| | Netzadresse: | 172.16.4.128 / 30 |
| | Broadcast: | 172.16.4.131 / 30 |
| **Verbindungsnetz** | Maske: | 255.255.255.252 |
| | GW-a: | 172.16.4.129 / 30 |
| | GW-b: | 172.16.4.130 / 30 |

# B Ganz knappe IOS-Anleitung

## B.1 IOS

Das *Internetwork Operation System* ist das Betriebssystem der Router der Firma Cisco. Die erste Version wurde bereits 1980 von Bill Yeager geschrieben und später von Cisco lizensiert.

IOS ist eine reine Konsolenanwendung, d.h. das Programm arbeitet **ähnlich** einer Unix-Shell und wird somit *ausschliesslich* über die Tastatur bedient. Die Ausgaben des IOS erfolgen dabei natürlich auch nur im reinen Textformat.

Wer bereits das Arbeiten mit z.b. der bash gewohnt ist, wir jedoch einen gravierenden Unterschied feststellen:

| Das IOS ist streng hierachisch aufgebaut. |
|---|

Das heisst, es gibt verschiedene *Ebenen*, von denen aus immer nur bestimmte Kommandos abgesetzt werden können. Diese Ebenen heissen innerhalb des IOS **Modes**.

Die Hierarchie der Modes sieht so aus:

**user-exec** : Von hier aus sind nur wenige Befehle möglich. Das Prompt sieht zum Beispiel so aus:

```
waraRouter>
```

**privileged-exec** : Von hier aus sind alle Administrationsbefehle zugänglich. Das Prompt im Priviledged-Exec-Mode sieht so aus:

```
waraRouter#
```

**configure** : Von hier aus sind Befehle zur Konfiguration des Routers zugänglich. Neues Prompt:

```
waraRouter(config)#
```

**sub-configuration-mode** : Je nach dem, was man konfigurieren möchte, gibt es Unter-Ebenen. Hier die wichtigsten:

**Line** : Von dieser Ebene aus wird der Konsolenzugang konfiguriert. Dieser kann über das serielle-I/F oder eine Telnet-Sitzung erfolgen. Beides muss einzeln konfiguriert werden (Bei Verwendung von quagga konfiguriert man den Router mit der Virtual-Shell: **vtysh**.). Das Prompt sieht beidesmal so aus:

```
waraRouter(config-line)#
```

**Interface** : Von dieser Ebene aus werden die physikalischen Schnittstellen des Routers konfiguriert. Das sind Bei echten CISCO-Routern:

- FastEthernet-Schnittstellen: fa0/0, fa0/1
- Weitverkehrs (WAN)-Schnittstellen: serial0/1/0, serial0/1/1

Dabei bezeichnen die Ziffern den Einschubschacht des Schnittstellenmoduls und auf dem Modul wiederum die Schnittstelle.

Bei einem quagga-Router heissen die Schnittstellen so, wie auf dem System auch, also: **eth0, eth1, eth2, lo**

Ach ja, das Prompt dieser Ebene ist so:

```
waraRouter(config-if)#
```

**Router** : Hier werden alle Routingfunktionen konfiguriert. Prompt:

```
waraRouter(config-router)#
```

## B.2 Beispiele für die Grundkonfiguration

Um einen fabrikneuen Router zu konfigurieren, muss man sich über die serielle RS232-Schnittstelle mit dem Router verbinden.

Auf dem Router ist diese Schnittstelle **hellblau** markiert. Als Terminal wird ein PC mit dem Programm hyperterm (windows) oder minicom (linux) verwendet.

Bei quagga genuegt es, **vtysh** aufzurufen.

Hier ein paar Beispiele für die Grundkonfiguration. Wir beginnen im user-exec-Mode und konfigurieren zuallererst die wichtigsten Passwörter. Die Konfiguration der Konsolenschnittstelle ist bei quagga nicht notwendig.

Mit dem Passwort `cisco` wird man sich später via Telnet oder Terminalprogramm am Router anmelden können, mit dem Passwort `class` wird man dann anschliessend vom user-exec-mode in den priviledged-exec-mode wechseln können. Bitte auf den Wara-Routern keine anderen Passwörter verwenden:

```
Router> enable                                      //in den priviledge-exec-mode wechseln
Router# configure terminal                          //in den config-mode wechseln
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#no ip domain-lookup                  //dns-lookup ausschalten.
Router(config)#hostname waraRouter                  //neuer Routername
Router(config)#enable secret class                  //das passwort fuer priviledged-exec setzen
waraRouter(config)#line console 0                   //konfiguration der konsolenschnittstelle
waraRouter(config-line)#password cisco              //login-passwort ist 'cisco'
waraRouter(config-line)#login                       //hiermit login scharfschalten
waraRouter(config-line)#line vty 0 4                //5 virtuelle konsolen (telnet) konfigurieren
waraRouter(config-line)#password cisco              //login-passwort für telnet
waraRouter(config-line)#login                       //und scharfschalten
waraRouter(config-line)#end                         //config-modus verlassen
waraRouter#copy running-config startup-config       //configuration im NVRAM sichern
```

Ab nun kann man sich nur mit dem Passwort 'cisco' am Router anmelden und muss das Passwort 'class' kennen, um in den priviledged-exec-mode zu wechseln.

Das Kommando `no ip domain-lookup` schaltet auf dem Router dns-Lookups aus. Sind diese eingeschaltet, interpretiert der Router jedes falsch eingegebene Kommando als dns-namen und versucht diesen aufzulösen, was ziemlich lange dauern kann.

Im Folgenden konfigurieren wir noch einen Begrüssungs-Text, der allen präsentiert wird, die eine Telnet-/Terminalsitzung zum Router starten:

```
waraRouter#configure terminal
waraRouter(config)#banner motd +
Enter TEXT message.  End with the character '+'.
```

```
FINGER WEG! NUR FUER ZUGANGSBERECHTIGTE!!!+
waraRouter(config)#end
waraRouter# copy run start
```

Falls die Router-Loggingmeldungen während des Arbeitens stören, kann man sie mit

```
waraRouter(config)#no logging console
```

im global-configuration-mode ausschalten.

# C Konfigurieren von Interfaces

Mit

```
waraRouter(config)#interface fa 0/0
```

gelangt man in den Konfigurationsmodus für das Fast-Ethernet-Interface 0/0. **fa** soll hier als Abkürzung für FAst-Ethernet stehen. 0/0 bedeutet: Modul0 / Interface0.

fa0/1 wäre demnach: Modul0 / Interface1.

Und bei quagga gibt es nur die System-Interfaces eth0, eth1, ... . Z.B.:

```
quaggaRouter(config)#interface eth1
```

Hier wieder ein Konfigurationsbeispiel für einen echten Router (ob sich bei einem quagga Router auch die serielle Schnittstelle konfigurieren lässt, ist dem Autor im Moment nicht bekannt. Auf jeden Fall müsste dazu das ppp-(P)rotokoll konfiguriert sein):

```
waraRouter(config)#interface fa0/0
waraRouter(config-if)#ip address 141.31.147.118 255.255.255.248
waraRouter(config-if)#no shutdown             //entspricht "UP"
waraRouter(config-if)#interface serial 0/1/0
waraRouter(config-if)#ip address 10.1.255.254 255.255.0.0
waraRouter(config-if)#no shutdown             //entspricht "UP"
waraRouter(config-if)#clock rate 64000        //das DCE-Ende gibt den Takt vor
waraRouter(config-if)#end
waraRouter#show ip interface brief            //alle ip-interfaces auflisten lassen
```

Bei seriellen Weitverkehrsverbindungen gibt die DCE-Schnittstelle den Datentakt vor. Die DTE-Schnittstelle synchronisiert sich auf diesen Takt.[2]

[2] DTE = Data terminal equipment, DCE = Data Circuit-terminating Equipment

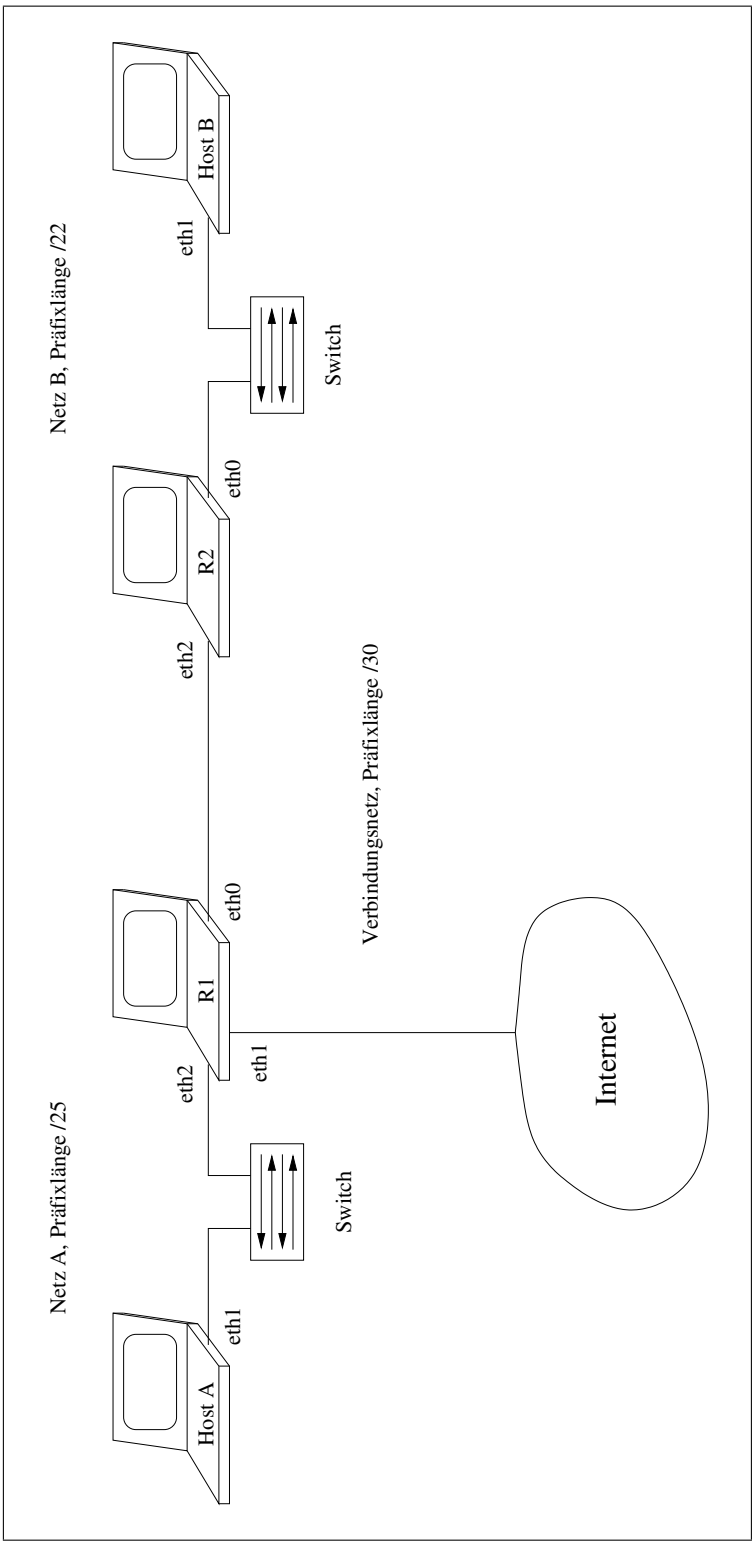


Abbildung 1: Ein einfaches Netz mit 2 Hops