Institut für Informatik und angewandte Mathematik
Universität Bern
Neubrückstrasse 10
CH-3012 Bern

Informatikprojekt

Aufbau eines IPv4/Ipv6-Experimentiernetzes

Thomas Bodenmann
bodenman@iam.unibe.ch

14. März 2001

Die bisherigen Forschungsarbeiten der Forschungsgruppe für Rechnernetze und verteilte Systeme beschränkten sich alle auf die aktuelle IP Version 4. Zukünftig wird aber das Internet Protokoll Version 6 (IPv6) eine zunehmende Bedeutung erfahren. Insbesondere für Multicast-Dienste und im Bereich der Mobilkommunikation bietet IPv6 signifikante Vorteile gegenüber IPv4. Ziel dieses Informatikprojekts war das Etablieren einer kombinierten IPv4/6-Testumgebung basierend auf Linux-Routern. Dabei war ein IPv4/IPv6-Experimentiernetz so aufzubauen, dass beide Protokollversionen nebeneinander existieren. Die in IPv6 zur Verfügung stehenden Mechanismen zur automatischen Konfiguration von Rechnern sollten beim Aufbau des IPv6-Experimentiernetzes genutzt werden. Zudem musste das IPv6-Testnetz per Tunneling an den weltweiten IPv6-Backbone (6Bone) angeschlossen werden. Ein weiterer Aspekt war der Test von existierenden IPv6-Testwerkzeugen wie `ping6`, `traceroute6`, etc. Das etablierte IPv4/6-Experimentiernetz soll nun als Basis für weiterführende Forschungsprojekte dienen.

Inhaltsverzeichnis

Abbildungsverzeichnis **5**

Tabellenverzeichnis **6**

1. Vorbereitungen **7**

1.1. Installation von Debian GNU/Linux 7

1.2. Anpassung und Erstellung des Betriebssystemkerns (Kernel) 7

1.2.1. Konfiguration des Kernels (v2.2.17) 8

1.2.2. Übersetzen des Kernels 9

1.2.3. Installation eines neuen Kernels 10

1.3. Netzwerkkartentreiber 10

2. IPv4-Experimentiernetz **11**

2.1. Konfiguration der Netzwerkkarten 11

2.2. Routing 12

2.2.1. Routing-Tabellen 13

2.3. Anbindung ans Internet 13

3. IPv6-Experimentiernetz **16**

3.1. 6Bone-Anschluss 17

3.2. Konfiguration der Netzwerkkarten 18

3.2.1. Host 18

3.2.2. Router 19

3.3. Tunnelendpunkt 21

3.4. Routing 21

3.4.1. Default-Route 21

3.4.2. Routing-Tabellen 21

3.5. `ifconfig`/`route` vs. `ip` 23

3.6. Analyse eines Router-Advertisement-Pakets (exemplarisch) 23

4. IPv6-Testwerkzeuge und -Applikationen **25**

4.1. Ping6, Traceroute6 . . . 25
4.2. Tcpdump . . . 25
4.3. Radvdump . . . 25
4.4. Ethereal . . . 26
4.5. Ttcp6 . . . 27
4.6. Mozilla . . . 28
4.7. Apache . . . 28

A. Konfigurationsdateien **30**

A.1. Allgemeine Dateien . . . 30
A.2. Netzwerkkonfiguration . . . 31
A.2.1. Optionen . . . 31
A.2.2. Netzwerkkartenkonfiguration und Routing . . . 31
A.2.3. Router-Advertisement-Daemon . . . 36
A.2.4. Kernelparameter für IPv6 . . . 38

Literaturverzeichnis **39**

Abbildungsverzeichnis

2.1. Topologie des IPv4-Experimentiernetzes 11

3.1. Topologie des IPv6-Experimentiernetzes 16

4.1. Ethereal 26

Tabellenverzeichnis

1.1. Vergabe der Hostnamen 7

2.1. IPv4-Routingtabellen 13

3.1. IPv6-Adressen der Hosts 17

3.2. Konfiguration der Router-Advertisement-Daemonen 20

4.1. Bandbreitenmessung mit `ttcp6` 27

1. Vorbereitungen

1.1. Installation von Debian GNU/Linux

Zum Aufbau der beiden koexistierenden Experimentiernetze standen sechs Rechner (PC's, Intel-Plattform) zur Verfügung. In einem ersten Schritt wurde auf allen Rechnern Debian GNU/Linux in der Version 2.2 („potato") direkt vom Debian-Mirror bei Switch (`ftp://sunsite.cnlab-switch.ch/mirror/debian/`) installiert. Bei den Netzwerkeinstellungen wurden die Hostnamen nach untenstehender Tabelle eingetragen. Die offiziellen IPv4-Adressen wurden nur während der Installation des Betriebssystems verwendet. Beim Aufbau des IPv4-Experimentiernetzes (siehe auch Kapitel 2) wurden diese durch private IP-Adressen ersetzt. Damit Debian vom Switch-Mirror installiert werden kann, sind zusätzlich folgende Angaben notwendig:

Hostname	IP-Adresse
`enterprise`	130.92.70.85
`challenger`	130.92.70.86
`atlantis`	130.92.70.87
`discovery`	130.92.70.88
`endeavour`	130.92.70.89
`columbia`	130.92.70.90

Tabelle 1.1.: Vergabe der Hostnamen

- IP-Adresse Gateway: `130.92.70.1`
- IP-Adresse Nameserver: `130.92.70.10`
- Domainname: `cnds.unibe.ch`
- HTTP-Proxy: `proxy.iam.unibe.ch`

1.2. Anpassung und Erstellung des Betriebssystemkerns (Kernel)

Nach der Installation von Debian musste ein den Bedürfnissen angepasster Kernel erstellt werden. Nachfolgend sind die dazu notwendigen Schritte beschrieben.

Es wird empfohlen, das Paket `kernel-package` zu installieren. Dieses Paket ermöglicht die automatische Erstellung von Debian-Paketen aus dem Kernelquellcode. Es ist zwar auch möglich, wie bei anderen Linux-Distributionen üblich, einen fertig erzeugten Kernel direkt zu installieren. Allerdings hätte dies zur Folge, dass der Paketmanager nicht mehr richtig über die installierten Dateien,

sowie die benutzte Kernelversion informiert wäre. Darüber hinaus lassen sich mit `kernel-package` leicht unterschiedlich konfigurierte Kernel aus der gleichen Version des Quellcodes erzeugen. Dies ist dann von Vorteil, wenn auf einem Rechner Kernel für andere Rechner erzeugt werden sollen oder mit verschiedenen Kernelkonfigurationen experimentiert werden soll. Die nachfolgend aufgeführten Befehle zur Kompilation und Installation des Kernels sind nur verwendbar, wenn das oben beschriebene Paket installiert ist. Welche Kernelparameter genau zu aktivieren sind, wird in Kapitel 1.2.1 beschrieben.

1.2.1. Konfiguration des Kernels (v2.2.17)

Um den Kernelquellcode zu konfigurieren, muss zunächst ins Basisverzeichnis des Kernelquellcodes (meist `/usr/src/linux/`) gewechselt werden. Danach muss durch Eingabe des Befels

```
debian:~# make-kpkg clean
```

sichergestellt werden, dass temporäre Dateien, die durch frühere Kernelkompilationen entstanden sein können, gelöscht werden. Anschliessend kann die Konfiguration des Kernels erfolgen. Die folgenden Überschriften entsprechen jenen der Kerneloptionen. Ein „`*`“ steht für „Option aktiviert“, ein „`M`“ für „Modul“. Die Darstellung der einzelnen Optionen orientiert sich an der Bildschirmausgabe, wie sie nach Eingabe des Befehls für die menübasierte Konfiguration (`make menuconfig`) des Kernel zu sehen ist.

Code maturity level options

```
[*] Prompt for development and/or incomplete code/drivers
```

Loadable module support

```
[*] Enable loadable module support
[*] Kernel module loader
```

Networking Options

```
<*> Packet socket
[*] Kernel/User netlink socket
[*] Routing messages
<*> Netlink device emulation
[*] Network firewalls
<*> Unix domain sockets
[*] TCP/IP networking
[*] IP: multicasting
[*] IP: advanced router
[*] IP: verbose route monitoring
[*] IP: firewalling
[*] IP: firewall packet netlink device
[*] IP: masquerading
--- Protocol-specific masquerading support will be built as modules.
```

```
[*] IP: ICMP masquerading
--- Protocol-specific masquerading support will be built as modules.
[*] IP: masquerading special modules support
<M> IP: ipautofw masq support (EXPERIMENTAL)
<M> IP: ipportfw masq support (EXPERIMENTAL)
<M> IP: ip fwmark masq-forwarding support (EXPERIMENTAL)
[*] IP: optimize as router not host
<*> IP: tunneling
[*] IP: aliasing support
<M> IP: Reverse ARP
[*] IP: Allow large windows (not recommended if <16Mb of memory)
<*> The IPv6 protocol (EXPERIMENTAL)
[*] IPv6: enable EUI-64 token format
[*] IPv6: disable provider based addresses
```

Network device support

```
[*] Network device support
<M> Dummy net driver support
Ethernet (10 or 100Mbit)  --->
    [*] Ethernet (10 or 100Mbit)
    [*] 3COM cards
    <M> 3c590/3c900 series (592/595/597) "Vortex/Boomerang"
    [*] EISA, VLB, PCI and on board controllers
    <M> EtherExpressPro/100 support
```

1.2.2. Übersetzen des Kernels

Nachdem der Kernel konfiguriert und die Konfiguration gesichert ist, kann der Kernel mit folgendem Befehl übersetzt werden:

```
debian:~# make-kpkg kernel_image --revision=anpassung.1
```

Über die Angabe des Arguments `kernel_image` wird `make-kpkg` mitgeteilt, alle Schritte durchzuführen, die notwendig sind, ein Debian-Paket zu erstellen, das sowohl den neuen Kernel als auch die ausgewählten Module enthält.

Revisionsnummern für Kernelpakete

Die Option `--revision=anpassung.1` bewirkt, dass die Revisionsnummer des zu erstellenden Debian-Pakets `anpassung.1` lautet. Die Verwendung dieser Option ist aus zwei Gründen sinnvoll:

1. Wenn Kernel für verschiedene Rechner zu erstellen sind, können diese unter Umständen unterschiedlich konfiguriert sein. Mit Revisionsnummern wie `host.1` oder `router.2` können die erstellten Debian-Pakte leicht voneinander unterschieden werden.

2. Ohne die Angabe einer Revisionsnummer verwendet `make-kpkg` die Versionsnummer `1.0`. Wenn das System regelmässig aktualisiert wird, kann es vorkommen, dass auf einer Installationsquelle (etwa einem Server mit Debian-Paketen) das gleiche Paket mit einer höheren

Versionsnummer zur Verfügung steht. Dies würde dazu führen, dass Paketmanagementprogramme wie `apt-get` oder `dselect` den eigenen, angepassten Kernel eventuell überschreiben würden, weil sie davon ausgehen, es handle sich bei den Paketen um eine aktualisierte Fassung des installierten Pakets.

1.2.3. Installation eines neuen Kernels

Das von `make-kpkg` erzeugte Debian-Paket wird in dem Verzeichnis abgelegt, in dem sich auch das Basisverzeichnis des Kernelquellcodes befindet. Unabhängig davon, ob ein angepasster Kernel als Debian-Paket erzeugt wurde oder ein Standard-Kernel der Distribution verwendet wird, kann dieser Kernel - wie jedes andere Debian-Paket - mit folgendem Befehl installiert werden:

```
debian:~# dpkg --install kernel-image-2.2.17_anpassung.1_i386.deb
```

Das im Paket enthaltene Installationsskript fragt den Benutzer, ob ein neuer Bootblock mit den bestehenden Konfigurationseinstellungen installiert werden soll. Diese ist mit `Yes` zu beantworten, damit der Bootloader LILO den neuen Kernel beim Start des Systems booten kann. Bei der Verwendung der im Anhang aufgeführten Datei `lilo.conf` wird zudem sichergestellt, dass sich der „alte“ Kernel durch Eingabe von `linuxold` am LILO-Prompt starten lässt.

1.3. Netzwerkkartentreiber

Als Netzwerkkarten stehen Modelle von 3Com (Serie „3c900“), resp. Intel („EtherExpress Pro 100“) zur Verfügung. Die Netzwerkartentreiber sind als Module kompiliert, damit auf allen Rechnern derselbe Kernel verwendet werden kann. Je nach Netzwerkkartenmodell muss zur Aktivierung der Karte in der Datei `/etc/modules` das entsprechende Modul durch Angabe des Namens (`3c59x`, resp. `eepro100`) geladen werden.

2. IPv4-Experimentiernetz

Den Rechnern des IPv4-Experimentiernetzes wurden private IP-Adressen aus dem Bereich $10.1.m.n$ (Klasse A) zugewiesen. Abbildung 2.1 zeigt die Topologie des IPv4-Experimentiernetzes. Die Nummern neben den einzelnen Verbindungen entsprechen den Werten $m.n$ des erwähnten Adressbereichs. Einzig `challenger`, welcher als Masquerading-Router das Experimentiernetz mit dem Internet verbindet, verfügt zusätzlich über eine offizielle IP-Adresse.

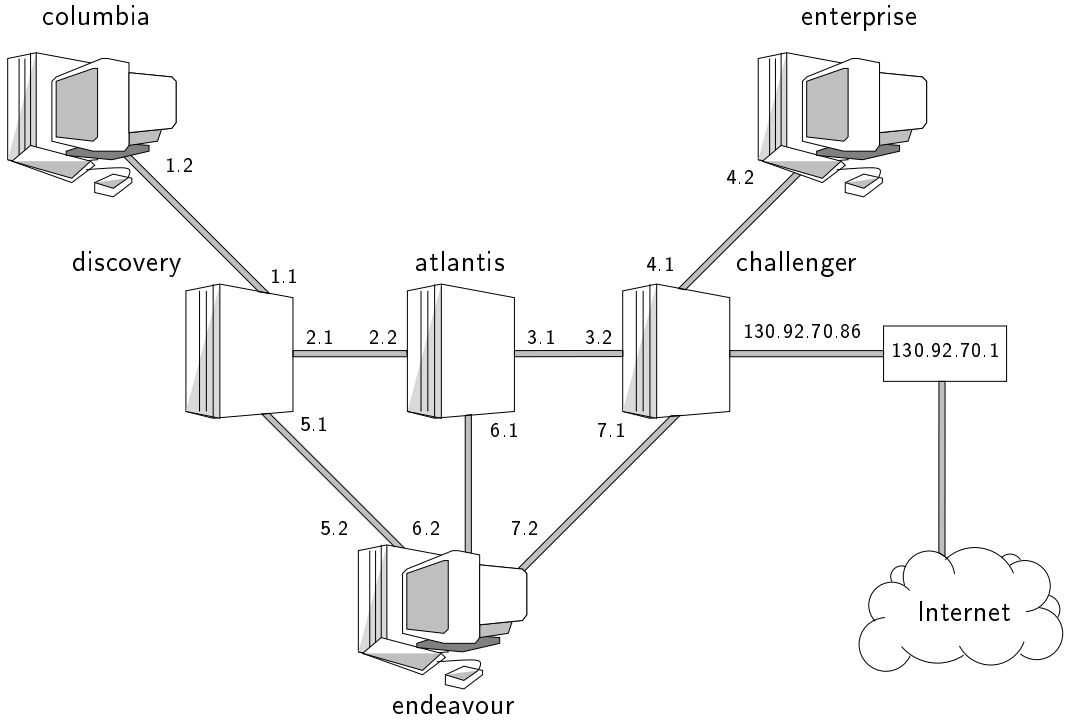

Abbildung 2.1.: Topologie des IPv4-Experimentiernetzes

2.1. Konfiguration der Netzwerkkarten

Alle netzwerkspezifischen Informationen werden bei Debian GNU/Linux in der Konfigurationsdatei `/etc/network/interfaces` eingetragen. Einträge in dieser Datei beginnen jeweils mit dem Schlüsselwort `iface` und mit der Bezeichnung des betreffenden Interfaces (z.B. `eth0`). Darauf folgt für Interfaces, die mit dem Internet-Protokoll benutzt werden, das Wort `inet`, sowie die Art der Konfiguration, hier `static`. Für die dynamische Zuweisung von IP-Adressen (DHCP, BOOTP) kann an dieser Stelle auch `dhcp` oder `bootp` als Konfigurationsmethode angegeben werden. Eine

vollständige Beschreibung der zur Verfügung stehenden Konfigurationsoptionen ist in der man-Page (`man interfaces`) nachzulesen. Die Aktivierung des Forwarding erfolgt durch einen Eintrag in der Datei `/etc/network/options`.

Sobald sich die Informationen in den beiden oben erwähnten Dateien befinden, können die Interfaces mit dem Befehl `ifup` aktiviert, resp. mit dem Befehl `ifdown` deaktiviert werden. Den Befehlen ist entweder der Name des zu (de-)konfigurierenden Interfaces oder der Parameter `-a` zu übergeben, dann werden alle definierten Interfaces aktiviert, resp. deaktiviert. Der Befehl `ifup` (`ifdown`) wird während des Systemsstarts (Systemstopps) vom Skript `/etc/init.d/networking` aufgerufen.

2.2. Routing

In der in Kapitel 2.1 erwähnten Datei `/etc/network/interfaces` lassen sich nebst netzwerkspezifischer Optionen auch Befehle angeben, welche während den verschiedenen Konfigurationsphasen der einzelnen Interfaces ausgeführt werden sollen. Dazu ist hinter den dafür reservierten Schlüsselwörtern (`up`, `pre-up`, `down` und `post-down`) der gewünschte Befehl einzugeben, wobei die einzelnen Schlüsselwörter mehrfach auftreten dürfen.

So können beispielsweise die Routen, die die einzelnen Interfaces „betreffen“ an dieser Stelle gesetzt werden. Untenstehend ist ein Ausschnitt aus der `interfaces`-Datei von `discovery` zu sehen:

```
iface eth1 inet static
        address 10.1.2.1
        netmask 255.255.255.0
        network 10.1.2.0
        broadcast 10.1.2.255
        gateway 10.1.2.2
        up route add -net 10.1.3.0 netmask 255.255.255.0 gw 10.1.2.2
        up route add -net 10.1.4.0 netmask 255.255.255.0 gw 10.1.2.2
        up route add -net 10.1.6.0 netmask 255.255.255.0 gw 10.1.2.2
        up route add -net 10.1.7.0 netmask 255.255.255.0 gw 10.1.2.2
        down route del -net 10.1.3.0 netmask 255.255.255.0 gw 10.1.2.2
        down route del -net 10.1.4.0 netmask 255.255.255.0 gw 10.1.2.2
        down route del -net 10.1.6.0 netmask 255.255.255.0 gw 10.1.2.2
        down route del -net 10.1.7.0 netmask 255.255.255.0 gw 10.1.2.2
```

2.2.1. Routing-Tabellen

In diesem Abschnitt sind die Routing-Tabellen aller Recher dargestellt. Die Tabellen stimmen mit der Ausgabe des Befehls `route -n` des in der Legende aufgeführten Rechners überein.

Netzwerk	Gateway	Netzmaske	Interface
10.1.4.0	0.0.0.0	255.255.255.0	eth1
10.1.5.0	10.1.4.1	255.255.255.0	eth1
10.1.6.0	10.1.4.1	255.255.255.0	eth1
10.1.7.0	10.1.4.1	255.255.255.0	eth1
10.1.1.0	10.1.4.1	255.255.255.0	eth1
10.1.2.0	10.1.4.1	255.255.255.0	eth1
10.1.3.0	10.1.4.1	255.255.255.0	eth1
0.0.0.0	10.1.4.1	0.0.0.0	eth1

(a) `enterprise`

Netzwerk	Gateway	Netzmaske	Interface
10.1.4.0	0.0.0.0	255.255.255.0	eth1
10.1.5.0	10.1.3.1	255.255.255.0	eth0
10.1.6.0	10.1.3.1	255.255.255.0	eth0
10.1.7.0	0.0.0.0	255.255.255.0	eth3
10.1.1.0	10.1.3.1	255.255.255.0	eth0
10.1.2.0	10.1.3.1	255.255.255.0	eth0
10.1.3.0	0.0.0.0	255.255.255.0	eth0
130.92.70.0	0.0.0.0	255.255.255.0	eth2
0.0.0.0	130.92.70.1	0.0.0.0	eth2

(b) `challenger`

Netzwerk	Gateway	Netzmaske	Interface
10.1.4.0	10.1.3.2	255.255.255.0	eth1
10.1.5.0	10.1.2.1	255.255.255.0	eth0
10.1.6.0	0.0.0.0	255.255.255.0	eth2
10.1.7.0	10.1.3.2	255.255.255.0	eth1
10.1.1.0	10.1.2.1	255.255.255.0	eth0
10.1.2.0	0.0.0.0	255.255.255.0	eth0
10.1.3.0	0.0.0.0	255.255.255.0	eth1
0.0.0.0	10.1.3.2	0.0.0.0	eth1

(c) `atlantis`

Netzwerk	Gateway	Netzmaske	Interface
10.1.4.0	10.1.2.2	255.255.255.0	eth1
10.1.5.0	0.0.0.0	255.255.255.0	eth2
10.1.6.0	10.1.2.2	255.255.255.0	eth1
10.1.7.0	10.1.2.2	255.255.255.0	eth1
10.1.1.0	0.0.0.0	255.255.255.0	eth0
10.1.2.0	0.0.0.0	255.255.255.0	eth1
10.1.3.0	10.1.2.2	255.255.255.0	eth1
0.0.0.0	10.1.2.2	0.0.0.0	eth1

(d) `discovery`

Netzwerk	Gateway	Netzmaske	Interface
10.1.4.0	10.1.7.1	255.255.255.0	eth2
10.1.5.0	0.0.0.0	255.255.255.0	eth0
10.1.6.0	0.0.0.0	255.255.255.0	eth1
10.1.7.0	0.0.0.0	255.255.255.0	eth2
10.1.1.0	10.1.5.1	255.255.255.0	eth0
10.1.2.0	10.1.6.1	255.255.255.0	eth1
10.1.3.0	10.1.6.1	255.255.255.0	eth1
0.0.0.0	10.1.7.1	0.0.0.0	eth2

(e) `endeavour`

Netzwerk	Gateway	Netzmaske	Interface
10.1.4.0	10.1.1.1	255.255.255.0	eth0
10.1.5.0	10.1.1.1	255.255.255.0	eth0
10.1.6.0	10.1.1.1	255.255.255.0	eth0
10.1.7.0	10.1.1.1	255.255.255.0	eth0
10.1.1.0	0.0.0.0	255.255.255.0	eth0
10.1.2.0	10.1.1.1	255.255.255.0	eth0
10.1.3.0	10.1.1.1	255.255.255.0	eth0
0.0.0.0	10.1.1.1	0.0.0.0	eth0

(f) `columbia`

Tabelle 2.1.: IPv4-Routingtabellen

2.3. Anbindung ans Internet

Um den Rechnern des IPv4-Experimentiernetzes Zugang zum Internet zu gewähren, wurde auf `challenger` IP-Masquerading eingerichtet. IP-Masquerading bezeichnet eine Form von „Network Address Translation“ (NAT), welche Rechnern mit privaten IP-Adressen mittels eines Masquerading-Routers den Zugriff auf das Internet ermöglicht.

Wenn der Masquerading-Router ein Datagramm eines Rechners des privaten Netzwerks erhält,

welches für das Internet bestimmt ist, so ersetzt er die (private) Source-Adresse durch seine (offizielle) IP-Adresse und vermerkt diese Modifikation in einer Tabelle. Dann wird das Datagramm zu demjenigen Host weitergeleitet, für den das Datagramm bestimmt ist. Dieser sendet allfällige Antwort-Datagramme zum Masquerading Router zurück. Der Router überprüft nun, ob das empfangene Datagramm zu einer „maskierten“ Verbindung gehört, modifiziert allenfalls die Destination-Adresse und sendet es an den entsprechenden Rechner im privaten Netzwerk.

Die eigentliche Einrichtung der Masquerade erfolgt mit dem Programm `ipchains`. Es dient in erster Linie dazu, Regeln zur Paketfilterung festzulegen und kann dem Kernel nebenher auch Anweisungen zum Masquerading mitteilen.

Zunächst kann dem Kernel mit `ipchains` mitgeteilt werden, wie lange eine Verbindung maskiert werden soll. Dazu ist das Programm wie folgt aufzurufen:

```
challenger:~# ipchains -M -S 7200 10 160
```

Die drei Werte entsprechen den Timeout-Werten in Sekunden für (1) normale TCP-Verbindungen (2 Stunden), (2) für beendete TCP-Verbindungen und (3) für UDP-Verbindungen. Danach kann das Masquerading für Pakete, die aus dem IPv4-Experimentiernetzwerk stammen, durch den folgenden Befehl aktiviert werden.

```
challenger:~# ipchains -A forward -s 10.1.0.0/255.255.0.0 -i eth2 -j MASQ
```

Nach diesem Schritt sollte das Masquerading für die meisten IP-basierten Netzwerkprotokolle funktionieren. Für einige Protokolle (z.B. FTP) ist es jedoch notwendig, zusätzliche Kernelmodule zu laden. Damit FTP über die Masquerade funktioniert, muss das Modul `ip_masq_ftp` geladen werden:

```
challenger:~# modprobe ip_masq_ftp
```

Damit die oben angebenen Schritte nicht jedesmal eingegeben werden müssen, wenn `challenger` gestartet wird, kann untenstehendes Skript unter dem Namen `/etc/init.d/masquerade` gespeichert werden.Die symbolischen Links in den einzelnen Startverzeichnissen für die verschiedenen Runlevels können mit folgendem Befehl erzeugt werden:

```
challenger:~# update-rc.d masquerade defaults 50 10
```

Datei: **/etc/init.d/masquerade**

```
#!/bin/sh
#
# Skript zum Starten des Masquerading

NET=10.1.0.0
MASK=255.255.0.0
IFACE=eth2

# falls Kernel kein Masquerading unterstuetzt, Skript abbrechen
test -f /proc/net/ip_masquerade || exit 0;

IPC=/sbin/ipchains

case "$1" in
    start)
        echo -n "Starting IP masquerading: "
        $IPC -M -S 7200 10 160
        $IPC -A forward -s $NET/$MASK -i $IFACE -j MASQ
        modprobe ip_masq_ftp
        echo "done."
        ;;
    stop)
        echo -n "Stopping IP masquerading: "
        $IPC -D forward -s $NET/$MASK -i $IFACE -j MASQ
        modprobe -r ip_masq_ftp
        echo "done."
        ;;
    restart)
        $0 stop;
        $0 start;
        ;;
    *)
        echo "Usage: /etc/init.d/masquerade {start|stop|restart}"
        exit 1
        ;;
esac

exit 0
```

3. IPv6-Experimentiernetz

Die Topologie des IPv6-Experimentiernetzes orientiert sich an jener des IPv4-Experimentiernetzes wie in Abbildung 3.1 zu sehen ist. Der selbe Aufbau wurde schon für Tests einer DiffServ-Implementierung [SB00] verwendet. Um die vollständige (global aggregierbare) IPv6-Adresse der einzelnen Rechner zu erhalten, muss vor die Nummern das Präfix 3ffe:2008:0::/48 gesetzt werden. Dies entpricht dem 6Bone-Adressraum der Universität Bern und wurde dem IAM von SWITCH zugewiesen.

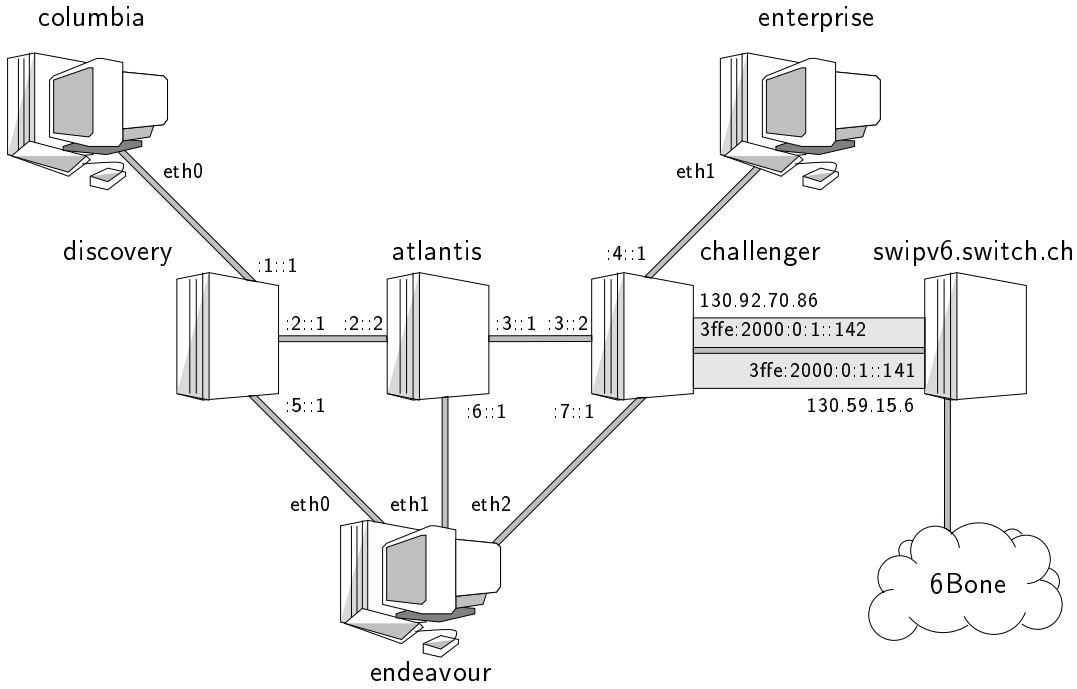

Abbildung 3.1.: Topologie des IPv6-Experimentiernetzes

Da die in IPv6 vorhandenen Möglichkeiten zur automatischen Adresskonfiguration ([NNS98],[TN98]) genutzt werden sollten, wurde eine Unterteilung der Rechner in Hosts und Router vorgenommen. Router sind `challenger`, `atlantis` und `discovery`. Als Hosts fungieren `enterprise`, `endeavour` und `columbia`.

Die unteren 64 Bit der IPv6-Adresse, die sog. „Interface ID“ [HD98b], der Netzwerkkarten der Hosts werden automatisch aus der MAC-Adresse der jeweiligen Netzwerkkarte generiert. Da die Netzwerkkarten der Router manuell konfiguriert werden müssen, wurde eine Durchnumerierung der Interface-ID's durchgeführt. Eine ausführliche Beschreibung der Konfiguration von Hosts und Routern ist in Kapitel 3.2 zu finden.

Host	Interface	IPv6-Adresse
columbia	eth0	3ffe:2008:0:1:210:4bff:fe67:f62/64
enterprise	eth1	3ffe:2008:0:4:210:4bff:fe6e:637f/64
endeavour	eth0	3ffe:2008:0:5:2a0:c9ff:feef:701a/64
	eth1	3ffe:2008:0:6:2d0:b7ff:fe40:884d/64
	eth2	3ffe:2008:0:7:2d0:b7ff:fe40:884c/64

Tabelle 3.1.: IPv6-Adressen der Hosts

3.1. 6Bone-Anschluss

Das IPv6-Experimentiernetz ist mittels eines IPv6-in-IPv4-Tunnels an den 6Bone angeschlossen. Der Tunnelendpunkt welcher dem IAM den Zugang zum 6Bone erlaubt, wird von SWITCH zur Verfügung gestellt. Zur Konfiguration des Tunnels wurden dem IAM folgende Daten übermittelt, wobei die in Klammern stehenden Werte von der Fachgruppe für Rechnernetze und verteilte System festgelegt wurden.

Tunnelendpunkt SWITCH

IPv4-Adresse : 130.59.15.6

IPv4-Hostname : swipv6.switch.ch

IPv6-Adresse : 3ffe:2000:0000:0001:0000:0000:0000:0141

IPv6-Prefix Length: 124

IPv6-Hostname : swipv6-t22.ipv6.switch.ch

Tunnelendpunkt IAM

IPv4-Adresse : (130.92.70.86)

IPv4-Hostname : (challenger.cnds.unibe.ch)

IPv6-Adresse : 3ffe:2000:0000:0001:0000:0000:0000:0142

IPv6-Prefix Length: 124

IPv6-Hostname : iam-unibe-swipv6.ipv6.switch.ch

3.2. Konfiguration der Netzwerkkarten

Auch für die Konfiguration von IPv6 kann die bereits in Kapitel 2.1 erwähnte Konfigurationsdatei (`/etc/network/interfaces`) verwendet werden. Die vorgenommene Unterteilung der Rechner in Hosts und Router macht sich bei der Konfiguration bemerkbar. Während sich die Konfiguration eines Hosts einfach gestaltet, ist bei der Konfiguration eines Routers etwas mehr Aufwand erforderlich. Nachfolgend sind die zur Konfiguration von Hosts und Routern notwendigen Schritte beschrieben.

3.2.1. Host

Die Konfiguration eines IPv6-Hosts unter Linux ist denkbar einfach: Es muss nur IPv6-Support im Kernel vorhanden sein! So wird beispielsweise die Link-lokale IPv6-Adresse der Netzwerkkarte `eth0` von `enterprise` automatisch aus deren 48 Bit MAC-Adresse generiert[1]. Das Präfix, mit welchem aus der Link-lokalen Adresse eine global aggregierbare Adresse gemacht werden kann, erhalten die Hosts per Router-Advertisement-Nachricht. Eine global aggregierbare IPv6-Adresse ist dann notwendig, wenn der betreffende Host nicht nur mit Rechnern kommunizieren will, die am gleichen Link angeschlossen sind: Die Link-lokalen IPv6-Adressen sind nicht routebar. Hat ein Host auf einem Interface, welches zur automatischen Adresskonfiguration vorgesehen ist, eine Router-Advertisement-Nachricht erhalten, wird das Präfix der link-lokalen Adresse durch das per Router-Avertisement erhaltene Präfix ersetzt. Danach besitzt ein Interface eine Link-lokale und eine global aggregierbare IPv6-Adresse. Die Konfiguration eines Interfaces kann beispielsweise mit folgendem Befehl eingesehen werden:

```
enterprise:~# ifconfig eth1

eth1      Link encap:Ethernet  HWaddr 00:10:4B:6E:63:7F
          inet addr:10.1.4.2  Bcast:10.1.4.255  Mask:255.255.255.0
          inet6 addr: fe80::210:4bff:fe6e:637f/10 Scope:Link
          inet6 addr: 3ffe:2008:0:4:210:4bff:fe6e:637f/64 Scope:Global
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1987862 errors:0 dropped:0 overruns:0 frame:0
          TX packets:985029 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          Interrupt:5 Base address:0xe400

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:3924  Metric:1
          RX packets:14 errors:0 dropped:0 overruns:0 frame:0
          TX packets:14 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
```

Zusätzlich müssen die Kernelparameter so angepasst werden, dass der Kernel Router-Advertisements akzeptiert. Dies kann mittels des Programms `sysctl` und der dazugehörenden Konfigurationsdatei `sysctl.conf` erfolgen. Diese Methode ist wesentlich praktischer, als wenn die Werte

[1] Die Abbildung von MAC-Adressen auf IPv6-Adressen ist in [HD98b] detailliert beschrieben.

mittels des `echo`-Befehls ins `/proc`-Verzeichnis geschrieben werden müssen. Hier die zur Konfiguration notwendigen Parameter:

```
# Autokonfiguration ein,
net.ipv6.conf.all.autoconf = 1
# Router-Advertisements akzeptieren,
net.ipv6.conf.all.accept_ra = 1
# aber keine Redirects.
net.ipv6.conf.all.accept_redirects = 0
# Forwarding aus!
net.ipv6.conf.all.forwarding = 0
# sende Konfigurationsanfragen
net.ipv6.conf.all.router_solicitations = 1
```

Diese Parameter sind in den aktuellen Kernelsourcen überhaupt nicht dokumentiert. So funktioniert beispielsweise die Autokonfiguration eines Hosts nicht, wenn IPv6-Forwarding aktiviert ist... Zu beachten ist zudem, dass am Ende der Datei `sysctl.conf` nicht die Unix-typische Leerzeile stehen darf, ansonsten gibt der Kernel eine Fehlermeldung aus, setzt aber trotzdem die angegeben Werte.

3.2.2. Router

Zur Konfiguration eines IPv6-Routers ist etwas mehr Aufwand erforderlich als bei der Konfiguration eines IPv6-Hosts. Den einzelnen Interfaces muss (im hier beschriebenen Szenario) eine global aggregierbare IPv6-Adresse zugewiesen werden. Dies, wie bei IPv4-Adressen, mit Hilfe der Datei `/etc/network/interfaces`. Für das Interface `eth0` von `discovery` sieht der Eintrag wie folgt aus:

```
iface eth0 inet6 static
        address 3ffe:2008:0:1::1
        netmask 64
```

Auch für die Router müssen die Kernelparameter entsprechend gesetzt werden:

```
# Autokonfiguration aus,
net.ipv6.conf.all.autoconf = 0
# keine Router-Advertisements und
net.ipv6.conf.all.accept_ra = 0
# keine Redirects akzeptieren.
net.ipv6.conf.all.accept_redirects = 0
# Forwarding ein!
net.ipv6.conf.all.forwarding = 1
# sende keine Konfigurationsanfragen
net.ipv6.conf.all.router_solicitations = 0
```

Router-Advertisement-Daemon

Damit die Autokonfiguration der Hosts funktioniert, muss auf den Routern des Netzwerks ein Daemon gestartet werden. Dieser Router-ADVertisement-Daemon („`radvd`“) beantwortet die Konfigurationsanfragen der Hosts (Router-Solicitations) mit sog. Router-Advertisement-Paketen. Diese

enthalten die zur automatischen Konfiguration notwendigen Daten, wie IPv6-Adress-Präfixe oder Informationen über die Default-Route.

Nach der Installation des Debian-Pakets (`ftp://ftp.debian.org/debian/dists/woody/main/binary-i386/net/radvd_0.5.0-1.deb`) muss im Verzeichnis `/etc/` die Datei `radvd.conf` erstellt werden, welche zur Konfiguration des Daemons verwendet wird. In dieser Konfigurationsdatei muss für jedes Interface, das Router-Advertisement-Pakete verschicken soll, ein entsprechender Eintrag vorhanden sein. Für das Interface `eth0` von `discovery` ist der Eintrag der folgenden Zeilen in `radvd.conf` nötig:

```
interface eth0
{
   AdvSendAdvert on;
   prefix 3ffe:2008:0000:0001::0/64
   {
      AdvOnLink on;
      AdvAutonomous on;
   };
};
```

Diese besagen, dass auf dem Interface `eth0` periodisch Router-Advertisement-Pakete mit dem Präfix 3ffe:2008:0:1::/64 verschickt werden sollen (`AdvSendAdvert on`). Die Option `AdvAutonomous on` bedeutet, dass das Präfix zur automatischen Adresskonfiguration verwendet werden kann. `AdvOnLink on` zeigt dem Host an, dass alle Hosts, welche das gleiche Präfix verwenden, über das Interface erreichbar sind, auf welchem er das Router-Advertisement-Paket empfangen hat.

Eine vollständige Beschreibung der zur Verfügung stehenden Konfigurationsoptionen ist in der man-Page (`man radvd.conf`) zu finden. Die Namen der einzelnen Optionen stimmen hierbei mit jenen der Spezifikation des Neighbor-Discovery Protokolls [NNS98] überein.

Eine zusammenfassende Auflistung der Konfiguration des Router-Advertisement-Daemons der einzelnen Router zeigt Tabelle 3.2.

Router	Interface	Präfix	`AdvSendAdvert`
`challenger`	eth0	3ffe:2008:0000:0003::0/64	off
	eth1	3ffe:2008:0000:0004::0/64	on
	eth3	3ffe:2008:0000:0007::0/64	on
`atlantis`	eth0	3ffe:2008:0000:0002::0/64	off
	eth1	3ffe:2008:0000:0003::0/64	off
	eth2	3ffe:2008:0000:0006::0/64	on
`discovery`	eth0	3ffe:2008:0000:0001::0/64	on
	eth1	3ffe:2008:0000:0002::0/64	off
	eth2	3ffe:2008:0000:0005::0/64	on

Tabelle 3.2.: Konfiguration der Router-Advertisement-Daemonen

Die Spalte `AdvSendAdvert` gibt den Wert der gleichnamigen Variable in der Datei `/etc/radvd.conf` an. Auch wenn die Variable auf den Wert „`off`“ gesetzt wird, können die Hosts ein Router-Advertisement-Paket mittels eines Router-Solicitation-Pakets anfordern.

3.3. Tunnelendpunkt

Wie in Abbildung 3.1 ersichtlich ist, bildet `challenger` den Tunnelendpunkt seitens des IAM. Ausgangspunkt für die Konfiguration eines IPv6-in-IPv4-Tunnels ist auch hier die Datei `/etc/network/interfaces`. Zur Konfiguration ist der Eintrag der folgenden Zeilen erforderlich:

```
iface swipv6tunl inet6 v4tunnel
        address 3ffe:2000:0000:0001:0000:0000:0000:0142
        netmask 124
        endpoint 130.59.15.6
        up ip route add 2000::0/3 via 3ffe:2000:0:1::141
```

Damit eine Konfiguration des Tunnels mittels der `interfaces`-Datei erfolgen kann, ist die Installation des Programms `ip` notwendig, welches sich im Paket `iproute` befindet.

3.4. Routing

Die Routen für IPv6 lassen sich nach der gleichen Methode wie jene für IPv4 setzen. Der einzige Unterschied besteht in der Verwendung des Befehls `ip`. Mehr Informationen zu `ip` sind in Kapitel 3.5 zu finden. Exemplarisch sei hier ebenfalls ein Ausschnitt aus der `interfaces`-Datei von `discovery` aufgelistet:

```
iface eth1 inet6 static
        address 3ffe:2008:0:2::1
        netmask 64
        up ip route add 3ffe:2008:0:3::0/64 via 3ffe:2008:0:2::2
        up ip route add 3ffe:2008:0:4::0/64 via 3ffe:2008:0:2::2
        up ip route add 3ffe:2008:0:6::0/64 via 3ffe:2008:0:2::2
        up ip route add 3ffe:2008:0:7::0/64 via 3ffe:2008:0:2::2
        up ip route add 2000::/3 via 3ffe:2008:0:2::2
```

3.4.1. Default-Route

Aus irgendwelchen Gründen, die wohl (vorerst) nur den Kernel-Programmierern vorenthalten bleiben, lässt sich die IPv6-Default-Route teilweise nicht korrekt setzen. Als „Workaround“ wird anstelle der Default-Route (::/0) die Route 2000::/3 verwendet. Man beachte hierzu auch die Konfigurationsdateien der einzelnen Rechner im Anhang.

3.4.2. Routing-Tabellen

Die IPv6-Routing-Tabellen sind aufgrund der „Lernfähigkeit“ des Kernels entsprechend kompliziert, die Eingabe des Befehls

```
challenger:~# route -n -A inet6
```

auf `challenger` erzeugt folgende Ausgabe:

```
Kernel IPv6 routing table
Destination                    Next Hop           Iface
::1/128                        ::                 lo
3ffe:2000:0:1::142/128         ::                 lo
3ffe:2000:0:1::140/124         ::                 swipv6tunl
3ffe:2008:0:1::/64             3ffe:2008:0:3::1   eth0
3ffe:2008:0:2::/64             3ffe:2008:0:3::1   eth0
3ffe:2008:0:3::2/128           ::                 lo
3ffe:2008:0:3::/64             ::                 eth0
3ffe:2008:0:4::1/128           ::                 lo
3ffe:2008:0:4::/64             ::                 eth1
3ffe:2008:0:5::/64             3ffe:2008:0:3::1   eth0
3ffe:2008:0:6::/64             3ffe:2008:0:3::1   eth0
3ffe:2008:0:7::1/128           ::                 lo
3ffe:2008:0:7::/64             ::                 eth3
2000::/3                       3ffe:2000:0:1::141 swipv6tunl
fe80::a01:302/128              ::                 lo
fe80::a01:401/128              ::                 lo
fe80::825c:4656/128            ::                 lo
fe80::2a0:c9ff:feef:8268/128   ::                 lo
fe80::2a0:c9ff:feef:976b/128   ::                 lo
fe80::2d0:b7ff:fe40:8410/128   ::                 lo
fe80::2d0:b7ff:fe40:8831/128   ::                 lo
fe80::/10                      ::                 eth0
fe80::/10                      ::                 eth1
fe80::/10                      ::                 eth2
fe80::/10                      ::                 swipv6tunl
fe80::/10                      ::                 eth3
200::1/128                     ff02::1            eth3
ff00::/8                       ::                 eth0
ff00::/8                       ::                 eth1
ff00::/8                       ::                 eth2
ff00::/8                       ::                 swipv6tunl
ff00::/8                       ::                 eth3
```

Die Spalten „`Flags`“, „`Metric`“, „`Ref`“ und „`Use`“ wurden dabei aus Platzgründen entfernt. Die Eingabe von

```
enterprise:~# route -n -A inet6
```

auf einem Host, in diesem Beispiel `enterprise`, erzeugt folgende Ausgabe:

```
Kernel IPv6 routing table
Destination                                Next Hop                   Iface
::1/128                                    ::                         lo
3ffe:2008:0:4:210:4bff:fe6e:637f/128       ::                         lo
3ffe:2008:0:4::/64                         ::                         eth1
fe80::210:4bff:fe6e:637f/128               ::                         lo
fe80::/10                                  ::                         eth1
ff00::/8                                   ::                         eth1
::/0                                       fe80::2a0:c9ff:feef:8268   eth1
```

Auffallend ist hier das Vorhandensein der Default-Route am Ende der Routing-Tabelle. Als Wert für den „`Next Hop`“ ist die Link-lokale Adresse der Netzwerkkarte von `challenger` angegeben, die `enterprise` mit `challenger` verbindet. Die Spezifikation des Neighbor-Discovery-Protokolls [NNS98] verlangt, dass Router-Advertisement-Pakete als Source-Adresse die Link-lokale IPv6-Adresse des Interfaces enthalten müssen, von welchem die Router-Advertisement-Nachricht gesendet wurde.

Zusätzlich enthalten die Routing-Tabellen beider Rechner (also Router und Host) Einträge für die in [HD98c] definierten Multicast-Adressen, resp. deren Präfix (ff00::/8).

3.5. `ifconfig/route` vs. `ip`

Das in Kapitel 3.3 erstmals erwähnte Programm `ip` ist ein vollständiger Ersatz für die beiden altbekannten Programme `ifconfig` und `route`. IPv6-Routen die mit `route` gesetzt wurden, lassen sich teilweise nicht mehr löschen. Dieser unangenehme Effekt trat bei der Verwendung von `ip` nicht auf; aus diesem Grund wird die Verwendung dieses Programms empfohlen. Zur Anzeige der Routingtabellen, seien es diejenigen von IPv4 oder IPv6, eigenen sich `route` und `ip` gleichermassen.

3.6. Analyse eines Router-Advertisement-Pakets (exemplarisch)

Unten ist die (fragmentierte) Textausgabe des dritten Pakets der in Abbildung 4.1 dargestellten Neighbor-Discovery-Pakete zu sehen. Anhand der einzelnen Paket-Fragmente kann das Router-Advertisement-Paket genau untersucht werden. So sind beispielsweise im Abschnitt `Internet Control Message Protocol v6` in Übereinstimmung mit der in Kapitel 3.2.2 als Beispiel aufgeführten Konfigurationsdatei `/etc/radvd.conf` von `discovery` die einzelnen Flags der Router-Advertisement-Nachricht zu sehen. Ebenso ist ersichtlich, dass das Paket als Source-Adresse die Link-lokale IPv6-Adresse des Interfaces `eth0` von `discovery` enthält.

Ethernet-Frame

```
Frame 3 (110 on wire, 110 captured)
    Arrival Time: Nov 15, 2000 17:02:09.0460
    Time delta from previous packet: 0.477593 seconds
    Frame Number: 3
    Packet Length: 110 bytes
    Capture Length: 110 bytes
Ethernet II
    Destination: 33:33:00:00:00:01 (33:33:00:00:00:01)
    Source: 00:a0:c9:b4:49:77 (00:a0:c9:b4:49:77)
    Type: IPv6 (0x86dd)
```

Internet Protocol Version 6

```
Internet Protocol Version 6
    Version: 6
    Traffic class: 0x00
    Flowlabel: 0x00000
    Payload length: 56
    Next header: ICMPv6 (0x3a)
    Hop limit: 255
    Source address: fe80::2a0:c9ff:feb4:4977
    Destination address: ff02::1
```

Internet Control Message Protocol v6

```
Internet Control Message Protocol v6
    Type: 0x86 (Router advertisement)
    Checksum: 0x2d12
    Cur hop limit: 64
    Flags: 0x00070800
        0... .... .... .... .... .... .... .... = Not managed
        .0.. .... .... .... .... .... .... .... = Not other
    Router lifetime: 1800
    Reachable time: 0
    Retrans time: 0
    ICMPv6 options
        Type: 0x03 (Prefix information)
        Length: 32 bytes (0x04)
        Prefix length: 64
        Flags: 0xc0ffffff
            1... .... = Onlink
            .1.. .... = Auto
        Valid lifetime: 0xffffffff
        Preferred lifetime: 0x00093a80
        Prefix: 3ffe:2008:0:1::
    ICMPv6 options
        Type: 0x01 (Source link-layer address)
        Length: 8 bytes (0x01)
        Link-layer address: 00:a0:c9:b4:49:77
```

Hexdump

Hier der Hexdump des ganzen Router-Advertisement-Pakets:

```
   0  3333 0000 0001 00a0 c9b4 4977 86dd 6000   33........Iw..‘.
  10  0000 0038 3aff fe80 0000 0000 0000 02a0   ...8:...........
  20  c9ff feb4 4977 ff02 0000 0000 0000 0000   ....Iw..........
  30  0000 0000 0001 8600 2d12 4000 0708 0000   ........-.@.....
  40  0000 0000 0000 0304 40c0 ffff ffff 0009   ........@.......
  50  3a80 0000 0000 3ffe 2008 0000 0001 0000   :.....?. .......
  60  0000 0000 0000 0101 00a0 c9b4 4977        ............Iw
```

4. IPv6-Testwerkzeuge und -Applikationen

Nachfolgend sind einige Standardwerkzeuge beschrieben, welche IPv6-Unterstützung bieten. Berücksichtigt wurden, mit einigen Ausnahmen, nur Programme, welche während der Projektarbeit in der Stable-Distribution („potato“) von Debian vorhanden waren. Falls nicht anders vermerkt, lassen sich die unten aufgeführten Programme mittels `apt-get install programmname` installieren. Aktualisierte Programmbibliotheken, die allenfalls zur Ausführung benötigt werden, werden mit diesem Befehl ebenfalls installiert. Zu beachten ist hier, dass vorhandene Pakete der Stable-Distribution nach der Installation von neuen Programmbibliotheken eventuell nicht mehr richtig funktionieren. Generell ist von einer gemischten Installation von Paketen der Stable- und der Unstable-Distribution abzuraten. In `/etc/apt/sources.list` sollten also nur Installationsquellen der Stable-Distribution stehen.

Detailliertere Informationen sind in den man-Pages oder in der Dokumentation der jeweiligen Programme zu finden. Die Dokumentation zu Programmen oder zu Paketen im Allgemeinen ist unter `/usr/doc/` abgelegt.

4.1. Ping6, Traceroute6

Die beiden Standardprogramme `ping6` und `traceroute6` sind bereits im Paket `netbase` enthalten, sie brauchen also nicht separat installiert zu werden. Die Programme bieten die gleiche Funktionalität wie ihre Vorgänger für IPv4.

Wenn Traceroute über einen Tunnel verwendet wird, werden einige Hops übersprungen. Weiterführende Erläuterungen dazu sind im Dokument „Tunnels over IP in Linux-2.2“ zu finden, welches in der Dokumentation zum Paket `iproute` enthalten ist.

4.2. Tcpdump

Das altbewährte Programm `tcpdump` liegt ebenfalls in einer IPv6-kompatiblen Version vor. Das entsprechende Debian-Paket ist jedoch nur in der unstable-Distribution enthalten, deshalb wurde im Rahmen dieses Projekts auf einen Test verzichtet.

4.3. Radvdump

Dem Router-Advertisement-Daemon liegt das Programm `radvdump` bei, mit welchem sich Router-Advertisement-Pakete untersuchen lassen. Das Programm ist sehr einfach gehalten, es beschränkt sich auf eine Textdarstellung der auf den Interfaces des Rechners ankommenden oder abgehenden Router-Advertisement-Pakete. Besser geeignet hierfür ist Ethereal. Siehe auch Kapitel 4.4.

4.4. Ethereal

Ein sehr nützliches Tool zur Analyse von Netzwerkverkehr ist `etherreal`, welches bereits eine fortgeschrittene IPv6-Unterstützung bietet. Es ist in der Lage, IPv6- und ICMPv6-Pakete zu untersuchen. Abbildung 4.1 zeigt die Analyse von einigen Neighbor-Discovery-Nachrichten, welche während des Bootvorgangs von `columbia` aufgenommen wurden.

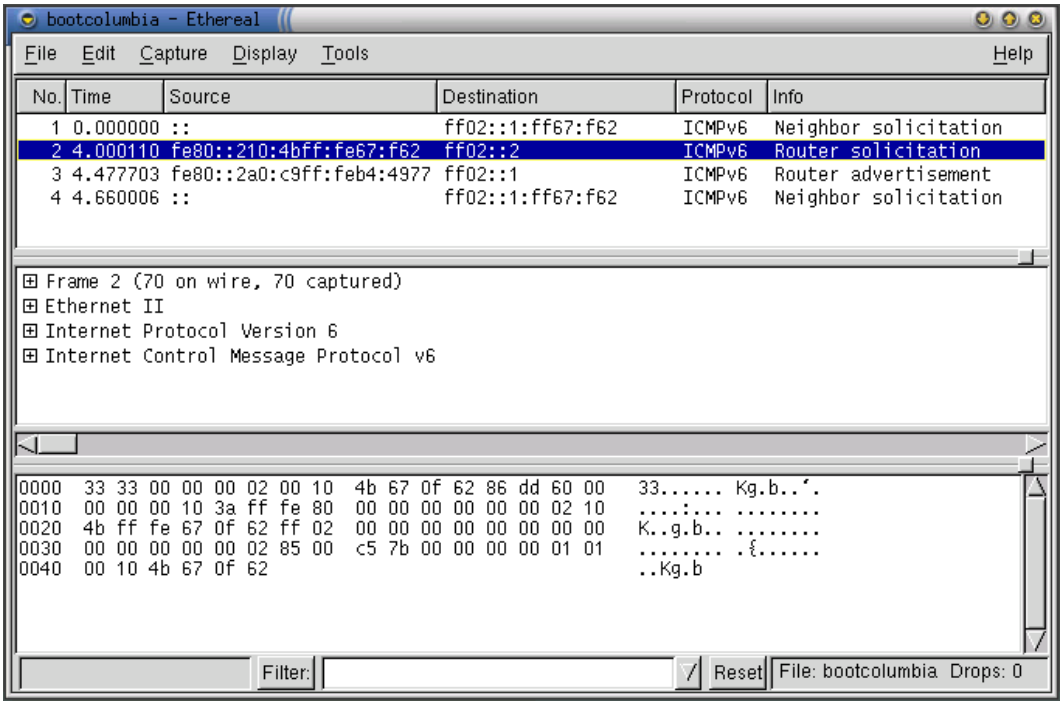

Abbildung 4.1.: Ethereal

Die dargestellten Information lassen sich auch in eine Textdatei ausgeben. Nachfolgend ist eine solche Textausgabe zu sehen, die Spalten „`No.`“ und „`Time`“ wurden dabei aus Platzgründen entfernt. Die Ausgaben lassen sich verschieden detailliert anfertigen, eine sehr detallierte Ausgabe eines der in Abbildung 4.1 gezeigten Pakete ist in Kapitel 3.6 abgedruckt.

```
Source                  Destination       Protocol Info
::                      ff02::1:ff67:f62  ICMPv6   Neighbor solicitation
fe80::210:4bff:fe67:f62 ff02::2           ICMPv6   Router solicitation
fe80::2a0:c9ff:feb4:4977 ff02::1          ICMPv6   Router advertisement
::                      ff02::1:ff67:f62  ICMPv6   Neighbor solicitation
```

Weitere Information zu Ethereal sind unter `http://www.ethereal.com` zu finden.

4.5. Ttcp6

Zur Bandbreitenmessung zwischen zwei Rechnern kann das Programm `ttcp6` verwendet werden. Dazu muss dieses auf dem Sender, resp. auf dem Empfänger mit verschiedenen Optionen gestartet werden. Das Programm muss zuerst auf dem Empfänger (hier `columbia`) mit

```
columbia:~$ ttcp6 -r
```

gestartet werden. Danach kann das Programm auf dem Sender, in diesem Beispiel `enterprise`, wie folgt gestartet werden:

```
enterprise:~$ ttcp6 -t -v -n1000 3ffe:2008:0:4:210:4bff:fe6e:637f
```

Die Zahl hinter der Option n steht für die Anzahl zu transferierender Blöcke. Als letztes Argument wird die IPv6-Adresse des Empfängers übergeben. Auf Senderseite erscheint dann nach Abschluss des Tests sinngemäss folgende Ausgabe:

```
ttcp6-t: buflen=8192, nbuf=1000, align=16384/0, port=5001
ttcp6-t: File-Descriptor 0x3 Opened
sockbufsize=65535,
# tcp sender -> 3ffe:2008:0:4:210:4bff:fe6e:637f #
ttcp6-t: 8192000 bytes in 1.000000 real seconds = 7.812 MB/sec +++
ttcp6-t: 8192000 bytes in 0.170000 cpu  seconds = 45.956 MB/cpu sec
ttcp6-t: 1000 I/O calls, 1.000 msec(real)/call, 0.170 msec(cpu)/call
ttcp6-t: 0.000000user 0.170000sys 0:01real 17.0%
ttcp6-t: buffer address 0x8050000
ttcp6-t: File-Descriptor  fd 0x3 Closed
ttcp6 done.
```

Die untenstehende Tabelle zeigt die der Ausgabe von `ttcp6` entnommenen Messwerte für verschiedene Werte von n.

Anzahl Blöcke (n)	Blockgrösse (Bytes)	Transferrate (MB/s)
1000	8192	7.812
5000	8192	13.021
10000	8192	9.766
50000	8192	11.161
100000	8192	11.004
1000000	8192	11.066

Tabelle 4.1.: Bandbreitenmessung mit `ttcp6`

Es existiert derzeit kein Debian-Paket; das Programm ist unter der URL `ftp://ftp.bieringer.de/pub/linux/IPv6/ttcp/ttcp+ipv6-2.tar.bz2` zu finden. Installationsanweisungen sind dem Archiv zu entnehmen.

4.6. Mozilla

Der Web-Browser Mozilla unterstützt sowohl IPv4 als auch IPv6. Nach der Installation kann der Browser durch die Eingabe des Befehls `mozilla` gestartet werden. Hier einige IPv6-URL's:

```
http://www.ipv6.eye-net.com.au/debian/
http://www.ipv6.bieringer.de/linux/IPv6/
```

Mit Webbrowsern ohne IPv6-Unterstützung lassen sich diese URL's nicht öffen („server not found"). Für IPv6-HTTP wird am IAM derzeit kein Proxy verwendet, Mozilla muss entsprechend konfiguriert werden.

Die währernd des Projekts installierte Version (M18) von Mozilla bietet (noch) keine fehlerfreie Unterstützung für das in [HCM99] spezifizierte Format für IPv6-Adressen in URL's. Deshalb wurde in die Datei `/etc/hosts` von `columbia` die IPv6-Adresse des HTTP-Servers (`enterprise`) eingetragen. Danach konnte von Mozilla aus ohne weitere Probleme auf den Server zugegriffen werden.

4.7. Apache

Nach dem Eintrag der Zeile

```
deb http://people.debian.org/~kitame/ipv6/ ipv6 potato
```

in der Datei `/etc/apt/sources.list` und dem Update der Paketdatenbank (`apt-get update`) kann eine IPv6-Version des HTTP-Server Apache mittels `apt-get install apache` installiert werden. Der HTTP-Daemon verlangt keine weiteren Konfigurationsschritte und wird unmittelbar nach der Installation automatisch gestartet.

Anhang

A. Konfigurationsdateien

Nachfolgend sind die zur Konfiguration der Rechner notwendigen Konfigurationsdateien aufgelistet. Falls der Rechnername nicht explizit angegeben ist, kann die betreffende Datei universell verwendet werden, d.h. für jeden Rechner kann die Datei ohne weitere Änderungen übernommen werden. Genauere Informationen zu den einzelnen Konfigurationsdateien sind in den man-Pages oder in [GOB00] zu finden.

A.1. Allgemeine Dateien

Datei: **/etc/lilo.conf**

```
# /etc/lilo.conf - See: ‘lilo(8)’ and ‘lilo.conf(5)’,
# --------------        ‘install-mbr(8)’, ‘/usr/share/doc/lilo/’,
#                        and ‘/usr/share/doc/mbr/’.

lba32
boot=/dev/hda
root=/dev/hda1
install=/boot/boot.b
map=/boot/map
# message=/boot/bootmess.txt
        prompt
        delay=100
        timeout=100
vga=normal
default=Linux

# Standardkonfiguration
image=/vmlinuz
        alias = 1
        label=Linux
        read-only

# falls mal etwas schiefgehen sollte....
image=vmlinuz.old
        alias = 2
        label=linuxold
        read-only
        optional
```

Datei: **/etc/modules**

```
# /etc/modules: kernel modules to load at boot time.
#
# This file should contain the names of kernel modules that are
# to be loaded at boot time, one per line.  Comments begin with
# a #, and everything on the line after them are ignored.
# Je nach Art der installierten Netzwerkkarten sind die folgenden
# Zeilen auszukommentieren, resp. die Karten durch das Entfernen
# von '#' zu aktivieren.

#3c59x
#eepro100
```

A.2. Netzwerkkonfiguration

A.2.1. Optionen

Datei: **/etc/network/options**

```
ip_forward=yes
spoofprotect=no
syncookies=no
```

A.2.2. Netzwerkkartenkonfiguration und Routing

enterprise

Datei: **/etc/network/interfaces**

```
# /etc/network/interfaces -- configuration file for ifup(8), ifdown(8)

iface lo inet loopback

iface eth1 inet static
        address 10.1.4.2
        netmask 255.255.255.0
        network 10.1.4.0
        broadcast 10.1.4.255
        gateway 10.1.4.1
        up route add -net 10.1.1.0 netmask 255.255.255.0 gw 10.1.4.1
        up route add -net 10.1.2.0 netmask 255.255.255.0 gw 10.1.4.1
        up route add -net 10.1.3.0 netmask 255.255.255.0 gw 10.1.4.1
        up route add -net 10.1.5.0 netmask 255.255.255.0 gw 10.1.4.1
        up route add -net 10.1.6.0 netmask 255.255.255.0 gw 10.1.4.1
        up route add -net 10.1.7.0 netmask 255.255.255.0 gw 10.1.4.1
        down route del -net 10.1.1.0 netmask 255.255.255.0 gw 10.1.4.1
        down route del -net 10.1.2.0 netmask 255.255.255.0 gw 10.1.4.1
        down route del -net 10.1.3.0 netmask 255.255.255.0 gw 10.1.4.1
        down route del -net 10.1.5.0 netmask 255.255.255.0 gw 10.1.4.1
        down route del -net 10.1.6.0 netmask 255.255.255.0 gw 10.1.4.1
        down route del -net 10.1.7.0 netmask 255.255.255.0 gw 10.1.4.1
```

challenger

Datei: **/etc/network/interfaces**

```
# /etc/network/interfaces -- configuration file for ifup(8), ifdown(8)

iface lo inet loopback

iface eth0 inet static
        address 10.1.3.2
        netmask 255.255.255.0
        network 10.1.3.0
        broadcast 10.1.3.255
        up route add -net 10.1.1.0 netmask 255.255.255.0 gw 10.1.3.1
        up route add -net 10.1.2.0 netmask 255.255.255.0 gw 10.1.3.1
        up route add -net 10.1.5.0 netmask 255.255.255.0 gw 10.1.3.1
        up route add -net 10.1.6.0 netmask 255.255.255.0 gw 10.1.3.1
        down route del -net 10.1.1.0 netmask 255.255.255.0 gw 10.1.3.1
        down route del -net 10.1.2.0 netmask 255.255.255.0 gw 10.1.3.1
        down route del -net 10.1.5.0 netmask 255.255.255.0 gw 10.1.3.1
        down route del -net 10.1.6.0 netmask 255.255.255.0 gw 10.1.3.1

iface eth0 inet6 static
        address 3ffe:2008:0:3::2
        netmask 64
        up ip route add 3ffe:2008:0:1::0/64 via 3ffe:2008:0:3::1
        up ip route add 3ffe:2008:0:2::0/64 via 3ffe:2008:0:3::1
        up ip route add 3ffe:2008:0:5::0/64 via 3ffe:2008:0:3::1
        up ip route add 3ffe:2008:0:6::0/64 via 3ffe:2008:0:3::1

iface eth1 inet static
        address 10.1.4.1
        netmask 255.255.255.0
        network 10.1.4.0
        broadcast 10.1.4.255

iface eth1 inet6 static
        address 3ffe:2008:0:4::1
        netmask 64

iface eth2 inet static
        address 130.92.70.86
        netmask 255.255.255.0
        network 130.92.70.0
        broadcast 130.92.70.255
        gateway 130.92.70.1

iface swipv6tunl inet6 v4tunnel
        address 3ffe:2000:0000:0001:0000:0000:0000:0142
        netmask 124
        endpoint 130.59.15.6
        up ip route add 2000::0/3 via 3ffe:2000:0:1::141

iface eth3 inet static
        address 10.1.7.1
        netmask 255.255.255.0
        network 10.1.7.0
        broadcast 10.1.7.255

iface eth3 inet6 static
        address 3ffe:2008:0:7::1
        netmask 64
```

atlantis

Datei: **/etc/network/interfaces**

```
# /etc/network/interfaces -- configuration file for ifup(8), ifdown(8)

iface lo inet loopback

iface eth0 inet static
        address 10.1.2.2
        netmask 255.255.255.0
        network 10.1.2.0
        broadcast 10.1.2.255
        up route add -net 10.1.1.0 netmask 255.255.255.0 gw 10.1.2.1
        down route del -net 10.1.1.0 netmask 255.255.255.0 gw 10.1.2.1

iface eth0 inet6 static
        address 3ffe:2008:0:2::2
        netmask 64
        up ip route add 3ffe:2008:0:1::0/64 via 3ffe:2008:0:2::1

iface eth1 inet static
        address 10.1.3.1
        netmask 255.255.255.0
        network 10.1.3.0
        broadcast 10.1.3.255
        gateway 10.1.3.2
        up route add -net 10.1.4.0 netmask 255.255.255.0 gw 10.1.3.2
        down route del -net 10.1.4.0 netmask 255.255.255.0 gw 10.1.3.2

iface eth1 inet6 static
        address 3ffe:2008:0:3::1
        netmask 64
        up ip route add 3ffe:2008:0:4::0/64 via 3ffe:2008:0:3::2
        up ip route add 2000::/3 via 3ffe:2008:0:3::2

iface eth2 inet static
        address 10.1.6.1
        netmask 255.255.255.0
        network 10.1.6.0
        broadcast 10.1.6.255
        up route add -net 10.1.5.0 netmask 255.255.255.0 gw 10.1.2.1
        up route add -net 10.1.7.0 netmask 255.255.255.0 gw 10.1.3.2
        down route del -net 10.1.5.0 netmask 255.255.255.0 gw 10.1.2.1
        down route del -net 10.1.7.0 netmask 255.255.255.0 gw 10.1.3.2

iface eth2 inet6 static
        address 3ffe:2008:0:6::1
        netmask 64
        up ip route add 3ffe:2008:0:5::0/64 via 3ffe:2008:0:2::1
        up ip route add 3ffe:2008:0:7::0/64 via 3ffe:2008:0:3::2
```

discovery

Datei: **/etc/network/interfaces**

```
# /etc/network/interfaces -- configuration file for ifup(8), ifdown(8)

iface lo inet loopback

iface eth0 inet static
        address 10.1.1.1
        netmask 255.255.255.0
        network 10.1.1.0
        broadcast 10.1.1.255

iface eth0 inet6 static
        address 3ffe:2008:0:1::1
        netmask 64

iface eth1 inet static
        address 10.1.2.1
        netmask 255.255.255.0
        network 10.1.2.0
        broadcast 10.1.2.255
        gateway 10.1.2.2
        up route add -net 10.1.3.0 netmask 255.255.255.0 gw 10.1.2.2
        up route add -net 10.1.4.0 netmask 255.255.255.0 gw 10.1.2.2
        up route add -net 10.1.6.0 netmask 255.255.255.0 gw 10.1.2.2
        up route add -net 10.1.7.0 netmask 255.255.255.0 gw 10.1.2.2
        down route del -net 10.1.3.0 netmask 255.255.255.0 gw 10.1.2.2
        down route del -net 10.1.4.0 netmask 255.255.255.0 gw 10.1.2.2
        down route del -net 10.1.6.0 netmask 255.255.255.0 gw 10.1.2.2
        down route del -net 10.1.7.0 netmask 255.255.255.0 gw 10.1.2.2

iface eth1 inet6 static
        address 3ffe:2008:0:2::1
        netmask 64
        up ip route add 3ffe:2008:0:3::0/64 via 3ffe:2008:0:2::2
        up ip route add 3ffe:2008:0:4::0/64 via 3ffe:2008:0:2::2
        up ip route add 3ffe:2008:0:6::0/64 via 3ffe:2008:0:2::2
        up ip route add 3ffe:2008:0:7::0/64 via 3ffe:2008:0:2::2
        up ip route add 2000::/3 via 3ffe:2008:0:2::2

iface eth2 inet static
        address 10.1.5.1
        netmask 255.255.255.0
        network 10.1.5.0
        broadcast 10.1.5.255

iface eth2 inet6 static
        address 3ffe:2008:0:5::1
        netmask 64
```

endeavour

Datei: **/etc/network/interfaces**

```
# /etc/network/interfaces -- configuration file for ifup(8), ifdown(8)

iface lo inet loopback

iface eth0 inet static
        address 10.1.5.2
        netmask 255.255.255.0
        network 10.1.5.0
        broadcast 10.1.5.255
        up route add -net 10.1.1.0 netmask 255.255.255.0 gw 10.1.5.1
        down route del -net 10.1.1.0 netmask 255.255.255.0 gw 10.1.5.1

iface eth1 inet static
        address 10.1.6.2
        netmask 255.255.255.0
        network 10.1.6.0
        broadcast 10.1.6.255
        up route add -net 10.1.2.0 netmask 255.255.255.0 gw 10.1.6.1
        up route add -net 10.1.3.0 netmask 255.255.255.0 gw 10.1.6.1
        down route del -net 10.1.2.0 netmask 255.255.255.0 gw 10.1.6.1
        down route del -net 10.1.3.0 netmask 255.255.255.0 gw 10.1.6.1

iface eth2 inet static
        address 10.1.7.2
        netmask 255.255.255.0
        network 10.1.7.0
        broadcast 10.1.7.255
        gateway 10.1.7.1
        up route add -net 10.1.4.0 netmask 255.255.255.0 gw 10.1.7.1
        down route del -net 10.1.4.0 netmask 255.255.255.0 gw 10.1.7.1
```

columbia

Datei: **/etc/network/interfaces**

```
# /etc/network/interfaces -- configuration file for ifup(8), ifdown(8)

iface lo inet loopback

iface eth0 inet static
        address 10.1.1.2
        netmask 255.255.255.0
        network 10.1.1.0
        broadcast 10.1.1.255
        gateway 10.1.1.1
        up route add -net 10.1.2.0 netmask 255.255.255.0 gw 10.1.1.1
        up route add -net 10.1.3.0 netmask 255.255.255.0 gw 10.1.1.1
        up route add -net 10.1.4.0 netmask 255.255.255.0 gw 10.1.1.1
        up route add -net 10.1.5.0 netmask 255.255.255.0 gw 10.1.1.1
        up route add -net 10.1.6.0 netmask 255.255.255.0 gw 10.1.1.1
        up route add -net 10.1.7.0 netmask 255.255.255.0 gw 10.1.1.1
        down route del -net 10.1.2.0 netmask 255.255.255.0 gw 10.1.1.1
        down route del -net 10.1.3.0 netmask 255.255.255.0 gw 10.1.1.1
        down route del -net 10.1.4.0 netmask 255.255.255.0 gw 10.1.1.1
        down route del -net 10.1.5.0 netmask 255.255.255.0 gw 10.1.1.1
        down route del -net 10.1.6.0 netmask 255.255.255.0 gw 10.1.1.1
        down route del -net 10.1.7.0 netmask 255.255.255.0 gw 10.1.1.1
```

A.2.3. Router-Advertisement-Daemon

challenger

Datei: **/etc/radvd.conf**

```
interface eth0
{
   AdvSendAdvert off;
   prefix 3ffe:2008:0000:0003::0/64
   {
      AdvOnLink on;
      AdvAutonomous on;
   };
};
interface eth1
{
   AdvSendAdvert on;
   prefix 3ffe:2008:0000:0004::0/64
   {
      AdvOnLink on;
      AdvAutonomous on;
   };
};
interface eth3
{
   AdvSendAdvert on;
   prefix 3ffe:2008:0000:0007::0/64
   {
      AdvOnLink on;
      AdvAutonomous on;
   };
};
```

atlantis

Datei: **/etc/radvd.conf**

```
interface eth0
{
   AdvSendAdvert off;
   prefix 3ffe:2008:0000:0002::0/64
   {
      AdvOnLink on;
      AdvAutonomous on;
   };
};
interface eth1
{
   AdvSendAdvert off;
   prefix 3ffe:2008:0000:0003::0/64
   {
      AdvOnLink on;
      AdvAutonomous on;
   };
};
interface eth2
{
   AdvSendAdvert on;
   prefix 3ffe:2008:0000:0006::0/64
   {
      AdvOnLink on;
      AdvAutonomous on;
   };
};
```

discovery

Datei: **/etc/radvd.conf**

```
interface eth0
{
   AdvSendAdvert on;
   prefix 3ffe:2008:0000:0001::0/64
   {
      AdvOnLink on;
      AdvAutonomous on;
   };
};
interface eth1
{
   AdvSendAdvert off;
   prefix 3ffe:2008:0000:0002::0/64
   {
      AdvOnLink on;
      AdvAutonomous on;
   };
};
interface eth2
{
   AdvSendAdvert on;
   prefix 3ffe:2008:0000:0005::0/64
   {
      AdvOnLink on;
      AdvAutonomous on;
   };
};
```

A.2.4. Kernelparameter für IPv6

Host

Datei: **/etc/sysctl.conf**

```
#
# /etc/sysctl.conf - Configuration file for setting system variables
# See sysctl.conf (5) for information.
#
#kernel.domainname = example.com
#net/ipv4/icmp_echo_ignore_broadcasts=1
net.ipv6.conf.all.autoconf = 1
net.ipv6.conf.all.accept_ra = 1
net.ipv6.conf.all.accept_redirects = 0
net.ipv6.conf.all.forwarding = 0
net.ipv6.conf.all.router_solicitations = 1
```

Router

Datei: **/etc/sysctl.conf**

```
#
# /etc/sysctl.conf - Configuration file for setting system variables
# See sysctl.conf (5) for information.
#
#kernel.domainname = example.com
#net/ipv4/icmp_echo_ignore_broadcasts=1
net.ipv6.conf.all.autoconf = 0
net.ipv6.conf.all.accept_ra = 0
net.ipv6.conf.all.accept_redirects = 0
net.ipv6.conf.all.forwarding = 1
net.ipv6.conf.all.router_solicitations = 0
```

Literaturverzeichnis

[Bra99] Torsten Braun. *IPnG: Neue Internet-Dienste und virtuelle Netze*. dpunkt, 1999.

[CD98] A. Conta and S. Deering. Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification. RFC 2463, December 1998.

[Cra98] M. Crawford. Transmission of IPv6 Pckets over Ethernet Networks. RFC 2464, December 1998.

[GOB00] John Goerzen, Ossama Othman, and Michael Bramer. *Debian GNU/Linux Guide*. LinuxLand International, 2000.

[HCM99] R. Hinden, B. Carpenter, and L. Masinter. Format for Literal IPv6 Adresses in URL's. RFC 2732, December 1999.

[HD98a] R. Hinden and S. Deering. Internet Protocol, Version 6 (IPv6) Specification. RFC 2460, December 1998.

[HD98b] R. Hinden and S. Deering. IP Version 6 Addressing Architecture. RFC 2373, July 1998.

[HD98c] R. Hinden and S. Deering. IPv6 Multicast Address Assignments. RFC 2375, July 1998.

[HOD98] R. Hinden, M. O'Dell, and S. Deering. An IPv6 Aggregatable Global Unicast Address Format. RFC 2374, July 1998.

[KD00] Olaf Kirch and Terry Dawson. *Linux Network Administrator's Guide, Second Edition*. O'Reilly, 2000.

[NNS98] T. Narten, E. Nordmark, and W. Simpson. Neighbor Discovery for IP Version 6 (IPv6). RFC 2461, December 1998.

[SB00] Günther Stattenberger and Torsten Braun. Performance Evaluation of a Linux DiffServ Implementation. Internal Report, Institut für Informatik, Universität Bern, Schweiz, September 2000.

[Tan98] Andrew S. Tanenbaum. *Computernetzwerke*. Prentice Hall, 1998.

[TN98] S. Thomson and T. Narten. IPv6 Stateless Address Autoconfiguration. RFC 2462, December 1998.