IPv6 - Adressklassen

Michael Dienert

16. November 2011

Inhaltsverzeichnis

Zusammenfassung

Folgender Text enthält eine knappe Beschreibung des Aufbaus von IPv6-Adressen und deren Verwendung in Zonendateien des DNS.

Nach der Beschreibung der verschiedenen Unicast- und Multicast-Adressen folgt eine kurze Einführung in das Format der unterschiedlichen IPv6-Zonendatei-Records.

Den Abschluss bilden eine Reihe von Übungen zur *Statless Autoconfiguration* und zum Namensdienst `bind9`.

Die in diesem Dokument enthaltenen Beschreibungen der IPv6-Adressen basieren auf der RFC 4291.

Kapitel 1

Wichtige Unterschiede zu IPv4

Zunächst seinen einige wichtige Unterschiede zu IPv4 erklärt. Das ist vor allem hinsichtlich der verschiedenen Begriffe notwendig.

Begriffe - IPv6 definiert einige aus der IPv4-Welt bekannte Begriffe um:

> **Subnet Prefix** - damit ist der Netzwerk-Teil einer IPv6-Adresse gemeint. Bei einer IPv4-Adresse nennt man diesen Teil Network-ID. Oft wird mit *Prefix* aber einfach nur eine Netzadresse bezeichnet.
>
> **Interface Identifier** - Die IPv6-Interface-ID entspricht unter IPv4 der Host-ID.
>
> **Site** - laut RFC 3587 ein "cluster of subnets/links", d.h. eine Site besteht aus mehreren Subnetzen einer Organisation. Am besten liesse sich *Site* wohl mit *Standort* (eines Netzwerks) übersetzen. Eine Site wäre dann z.B. ein Unternehmens- oder Campusnetzwerk.
>
> **Link** - entspricht einem IPv4-Subnet.

Adressraum - IPv6-Adressen sind 128 bit lang. Der damit mögliche Adressraum ($340282366920938463463374607431768211456 = 3.40 \cdot 10^{38}$ Adressen) ist so gross, dass man eine Reihe von Vereinfachungen gegenüber IPv4 einführen konnte:

- Der riesige Adressraum macht NAT (Net-Address-Translation) überflüssig
- Da Adressen nicht mehr gespart werden müssen, brauchen Subnetze nicht mehr so klein als möglich genacht zu werden und man kann mit einheitlichen Subnetz-Masken arbeiten:

> Unicast IPv6-Adressen haben bis auf wenige Ausnahmen einen **festen** Präfix von /64.

Hexadezimale Schreibweise - Die unsägliche IPv4-Dezimalnotation (*Dotted Decimal Notation*) wurde endlich zugunsten einer Darstellung im Hexadezimalformat verworfen:

- die 128 Bit der Adresse werden in 8 Blöcke zu je 16 Bit unterteilt, die durch einen Doppelpunkt getrennt werden. Die 16 Bit eines Blocks werden mit 4 Hexadezimalziffern dargestellt [1] .
- führende Nullen in einem Block dürfen weggelassen werden.
- ein oder mehrere aufeinanderfolgende Blöcke, die nur aus Nullen bestehen dürfen gestrichen und durch '::' ersetzt werden. Pro Adresse darf das aber nur genau einmal gemacht werden!

Interface-Adressen - normalerweise hat ein physikalisches Interface nur eine IPv4-Adresse. In Ausnahmefällen, weist man einem Interface mehrere IPv4-Adressen zu (Subinterface, virtuelles Interface). Z.B bei einem Routerinterface, das mehrere VLANs verbinden soll. Bei IPv6 wird diese Ausnahme zur Regel:

Einem physikalischen Interface kann eine beliebige Zahl gleichberechtigter IPv6-Adressen zugewiesen werden.

Fragmentierung - Fragmentiert wird nur im äussersten Notfall und auch nur *ausschliesslich* auf der Sendeseite. Dabei wird die Pfad-MTU *vorab* mit *Path MTU Discovery* [2] ermittelt.

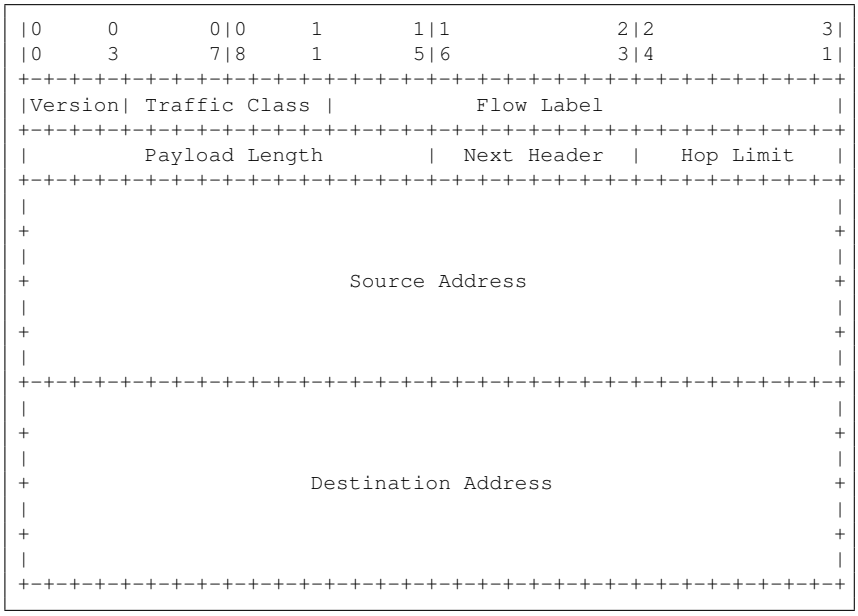
Eine Fragmentierung unterwegs wie bei IPv4 vorgesehen findet niemals statt.

einfacher Header - Da im Standardfall nicht fragmentiert wird, ist der IPv6-Header gegenüber dem IPv4-Header einfach aufgebaut: die für die Fragmentierung zuständigen Felder entfallen.

Mit den beiden 128bit IPv6-Adressen wird der Header somit 40Bytes lang. Hier die Original-IPv6-Header-Format-Darstellung aus der RFC 2460 (die Bitnummerierung stammt vom Autor dieses Dokuments und ist an RFC4291 angelehnt):

[1] Jede Hexadezimalziffer repräsentiert je 4 Bit. Für solch ein Quartett ist auch die Bezeichnung *Nibble* üblich.

[2] Das Verfahren Path MTU Discovery arbeitet mit *ICMPv6 Packet Too Big* - Meldungen die immer dann versandt werden, wenn ein Paket unterwegs auf eine Strecke gelangt, deren MTU kleiner als die Paketgrösse ist. Da die *PacketTooBig*-Meldung die MTU des Engpasses enthält, kann der Absender der Pakete die Paketgrösse daran anpassen.



Next Header Erwähnenswert ist noch das Feld *Next Header*. Der IPv6-Header besitzt keine Optionen wie der IPv4-Header mehr. Stattdessen werden die Optionen in einem Erweiterungs-Header untergebracht, der wiederum selbst auf einen Next Header zeigen kann (verkettete Liste).

Der letzte Header in dieser Liste enthält im Next-Header-Feld die Schlüsselnummer des transportierten Protokolls und entspricht somit gleichzeitig dem IPv4-Feld "*Protocol*". Im Anhang D.1 sind die möglichen Werte von *Next Header* aufgelistet.

Erweiterte Mulitcast-Adressen, Verzicht auf Broadcasts - Unter IPv6 ist das Konzept der Multicast-Adressen so weit verfeinert worden, dass Broadcasts überflüssig wurden. Broadcasts werden unter IPv6 also nicht mehr verwendet.

Kapitel 2

IPv6 Adressen

2.1 Geltungsbereiche

Bei IPv6 werden eine Vielzahl von Geltungsbereichen (Scopes) unterschieden. Die meisten davon werden im Zusammenhang mit *Multicast-Adressen* verwendet.
Die folgenden drei Bereiche sind jedoch besonders wichtig, das sie für alle Adressarten gelten:

Global Scope - Globaler Geltungsbereich. Adressen aus diesem Bereich werden im gesamten Internet6 geroutet.

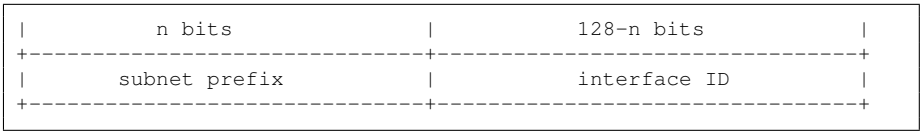
Site-Local Scope - Adressen mit einem Site-Lokalen Geltungsbereich werden nur innerhalb einer *Site* geroutet. Dabei lässt sich der Begriff Site leider nicht exakt definieren. Man kann aber Knoten in einem *begrenzten Netzwerk* Site-Lokale Adressen vergeben, womit sichergestellt ist, dass diese Knoten von ausserhalb dieses Netzwerks nicht erreicht werden können.

Link-Local Scope - Im IPv6-Sprachgebrauch steht *Link* für *Subnet*. D.h., Link-Local-Adressen können nur innerhalb eines Subnets verwendet werden und werden **nicht** geroutet.

Eine vollständige Übersicht finden Sie in Anhang D.2.

2.2 Unicast Adressen

Allgemein haben Unicast-Adressen folgendes Format:



2.2.1 Modified EUI-64-Format

In den meisten Fällen wird die allgemeine Form einer Unicast-Adressse eingeschränkt:

Bei allen Adressen, die nicht mit 000 beginnen muss die Interface-ID 64 Bit lang und im *Modified EUI-64-Format* sein.

Abhängig von der **Hardwareadresse** des Knotens gibt es nun mehrere Möglichkeiten, Modified EUI-64-Interface-Identifier zu erzeugen:

Hardwareadresse im IEEE EUI-64 Format - um vom Standard IEEE EUI-64 Format ins modifizierte Format zu gelangen, genügt es das **"u-bit"** (universal/local-Bit) zu *invertieren*.

Die FireWire-Schnittstelle (ieee 1394) verwendet z.B. das EUI-64-Format als Hardwareadresse, so dass einem FireWire-Port einfach eine IPv6-Adresse zugewiesen werden kann. RFC 3146 beschreibt eine Übertragung von IPv6-Paketen über IEEE 1394 Netzwerke.

Hardwareadresse mit IEEE 802 48-bit Macadressen -

- eine 48-bit-Macadresse wird durch Einfügen von `0xFFFE` auf 64 bit Länge gebracht
- das universal/local-bit wird gemäss Tabelle 2.1 gesetzt.

Ein Beispiel für ein Interface mit 48-Bit-Mac-Adresse:

```
|0      0         1|1               3|3               4|
|0      6         5|6               1|2               7|
+----------------+----------------+----------------+
|cccccc0gcccccccc|cccccccc mmmmmmmm|mmmmmmmmmmmmmmmmmmmm|
+----------------+----------------+----------------+
```

Daraus wird ein (global gültiger) Interface Identifier abgeleitet:

```
|0      0         1|1               3|3               4|4               6|
|0      6         5|6               1|2               7|8               3|
+----------------+----------------+----------------+----------------+
|cccccc1gcccccccc|cccccccc11111111|11111110mmmmmmmm|mmmmmmmmmmmmmmmmmmmm|
+----------------+----------------+----------------+----------------+
```

c=bit der Firmen-ID m=bit vom Hersteller vergeben (Manufacturer) g=individual/group-bit

Das hier auf 1 gesetzt Bit 06 ist das u-bit.

Es muss die in der Tabelle 2.1 gezeigten Werte haben.

Scope	u-bit
global	1
local	0

Tabelle 2.1: universal/local-bit

Grund für diese Regel: angenommen Sie als Administrator möchten manuell lokale Adressen vergeben. Mit ubit=0 (lokal) erhalten Sie für die Interface-Ids z.b. die Folge:

```
0:0:0:1, 0:0:0:2, 0:0:0:3, ...
```

mit ubit=1 würden Sie folgende, umständliche Adressen erhalten:

```
0200:0:0:1, 0200:0:0:2, 0200:0:0:3, ...
```

2.2.2 Link-Local Unicast-Adressen

Link-Local Adressen werden für eine Vielzahl von internen IPv6-Aufgaben verwendet. Sobald ein Interface für IPv6 konfiguriert wird, wird das Interface sich automatisch selbst eine Link-Local Adresse aus dem Bereich fe80::/64 vergeben.

Der gesamte Adressbereich fe80::/10 ist für Link-Local Adressen reserviert. Interfaces erhalten jedoch nur eine Adresse aus dem Bereich fe80::/64

Der Bereich fe80::/10 ergibt folgende Bereiche:

fe8x	Link-Local; zur Zeit in Verwendung
fe9x	Link-Local; noch nicht in Verwendung
feax	Link-Local; noch nicht in Verwendung
febx	Link-Local; noch nicht in Verwendung

Tabelle 2.2: **Link-Local**-Unicast-Adressbereiche

2.2.3 Site-Local Unicast-Adressen

Site-Local-Adressen entsprechen den aus der IPv4-Welt bekannten, privaten Adressklassen [1] .
Wie die IPv4-Pendants sie auch diese Adressen problematisch, da sie u.U. nicht eindeutig vergeben werden. Das führt z.B. zu Problemen beim Zusammenschluss von Netzen. Aus diesem Grund wurden die Site-Local-Unicast-Adressen abgekündigt.

fec0::/10 (Site Local) - dieser Bereich sollte nicht mehr verwendet werden. Ursprünglich sollte jeder Adressen aus diesem Bereich für lokale Aufgaben verwenden.

Mit fec0::/10 ergeben sich folgende Bereiche:

fecx	Site-Local, abgekündigt
fedx	Site-Local, abgekündigt
feex	Site-Local, abgekündigt
fefx	Site-Local, abgekündigt

Tabelle 2.3: **Site-Local**-Unicast-Adressbereiche

fc00::/8 (Unique Local) - Aus diesem Bereich sollen in Zukunft von einer zentralen Verwaltungsstelle Subnetze mit einem Präfix von /48 vergeben werden. Allerdings gibt es diese Verwaltungsstelle noch nicht.

[1] 10.0.0.0/8, 172.16.0.0/12 und 192.168.0.0/16

fd00::/8 (Unique Local) - Aus diesem Bereich kann man sich beliebig /48-er-Netzadressen (Netzadresse = Prefix) auswählen. Da der Bereich gross ist, ist die Wahrscheinlichkeit, dass zwei Sites das gleiche Subnetz haben (den gleichen Prefix haben), klein.

Adressen aus den Bereichen fc00::/8 und fd00::/8 nennt man **Unique-Local Adressen**. Da globale Adressen frei erhältlich sind, sollte man Unique-Local Adressen nur in Notfällen verwenden.

2.2.4 Global Scope Unicast-Adressen

Für global geroutete Adressen wurde mit der RFC 2374 der Bereich 2000::/3 (binär: alle Adressen, die mit 001 beginnen) vorgesehen. In der RFC 3587 wird der Bereich globaler Adressen allgemeiner definiert, so dass in Zukunft von der IANA evtl. noch weitere, bisher ungenutzte Bereiche des IPv6-Adressraums für gobale Adressen reserviert werden können.

Alle globalen Unicast-Adressen, **die nicht mit 000 beginnen** müssen folgendes Format haben:

```
|         n bits           | 64-n bits |       64 bits               |
+--------------------------+-----------+-----------------------------+
| global routing prefix    | subnet ID |       interface ID         |
+--------------------------+-----------+-----------------------------+
```

Man darf sich also nicht darauf verlassen, dass auch in Zukunft alle globalen Adressen mit binär 001 beginnen.

Im Moment ist das aber noch der Fall und die von der IANA vergebenen Adressen haben folgendes Format:

```
| 3 |     45 bits          |  16 bits  |       64 bits               |
+---+----------------------+-----------+-----------------------------+
|001|global routing prefix| subnet ID |       interface ID          |
+---+----------------------+-----------+-----------------------------+
```

Etwas anders notiert ergeben sich folgende Bereiche:

2xxx	global Unicast
3xxx	global Unicast

Tabelle 2.4: Bereiche global routebarer Adressen

2.2.4.1 Ein Beispiel

Im Beispiel wollen wir folgende Adresse analysieren:

```
2001:0db8:3c4d:0015:0000:0000:1a2f:1a2b
```

Zunächst betrachten wird nur die ersten 4 Bits:

```
Hex:      2    0    0    1    : . . . .
Binaer:   0010 0000 0000 0001 : . . . .
```

Die Adresse beginnt binär mit `001`, wie das von der IANA z.Zt. vorgeben wird.
Die ersten 32 bit in Hex sind:

```
2001:0db8
```

Das Netz (in IPv6 sagt man *der Präfix*) `2001:0db8::/32` ist von der APNIC [2] für Dokumentationszwecke vorgesehen worden. Damit ist der Präfix (das Subnet)

```
2001:0db8:3c4d::/48
```

der **Global Routing Präfix** unserer *Site*.
Die nächsten beiden Oktette - `0x0015` - sind die sog. *Subnet ID*. Innerhalb unserer Site können wir also $2^{16} = 65536$ Subnetze betreiben.
Die höchstwertigen 64 Bit stellen schliesslich die *Interface-ID* dar. Bemerkenswert ist hier, dass jedes der 65536 Subnetze *4 Milliarden mal so gross* ist wie das gesamte Internet4!

2.3 Multicast Adressen

Von Anfang an waren *Multicast Adressen* zentraler Bestandteil der IPv6 Spezifikation.
Das IPv6-Multicast-Konzept wurde jedoch gegenüber den Multicast-Adressen von IPv4 stark verfeinert, und unter IPv6 übernehmen Multicast-Adressen sogar die Funktion von Broadcasts:

IPv6 verwendet keine Broadcasts mehr

2.3.1 Multicast-Format

Multicast-Adressen haben folgendes Format (vgl. RFC 4291):

```
|   8    |  4 |  4 |                  112 bits                   |
+------ -+----+----+---------------------------------------------+
|11111111|flgs|scop|                  group ID                   |
+--------+----+----+---------------------------------------------+
```

flgs - Flag-Nibble mit dem Format

```
+-+-+-+-+
|0|R|P|T|
+-+-+-+-+
```

[2] Asia Pacific Network Information Centre, das ist die Regional Internet Registry (RIR) für die Region Asien und den Pazifik.

Wichtig ist hier das Flag T: T=0 bedeutet, dass die Multicast-Adresse "Well-Known", also von der IANA fest vergeben wurde. Entsprechend markiert T=1 eine dynamisch vergebene Mulitcast-Adresse (ähnlich wie Portadressen kleiner oder grösser 1024).

scop - Das ist das Scope-Nibble. Der binäre Wert verweist auf den Gültigkeitsbereich der Multicast-Adresse.

Ein Beispiel: `scop = 0101 = 5` : Der Geltungsbereich der Multicast-Adresse ist *Site-Local*.

Für eine vollständige Übersicht der Scope-Bereiche siehe Anhang D.2.

2.4 Wichtige Multicast-Adressen

Folgende Multicast-Adressen sind besonders wichtig und deshalb an dieser Stelle aufgeführt:

All Nodes - mit den beiden Adressen:

```
ff01::1     //scope=1; interface-local
ff02::1     //scope=2; link-local
```

lassen sich **alle** IPv6-Rechner (Nodes) im lokalen Scope erreichen.

All Routers - die All-Routers-Adressen sind:

```
ff01::2     //scope=1; interface-local
ff02::2     //scope=2; link-local
ff05::2     //scope=5; site-local
```

Solicited Node - Diese Multicast-Adresse spielt eine wichtige Rolle bei der Auflösung der Layer2-Adresse (MAC-Adresse). Wie diese Adressen eingesetzt und gebildet werden ist in einer Übung in Kap. 4.0.1.1 beschrieben.

2.5 Loopback Adresse

Die IPv6-Loopback-Adresse hat den Wert **::1** . Sie darf ausschliesslich für das Loopback-Interface verwendet werden.

2.6 Unspecified Adresse

Die Adresse **0:0:0:0:0:0:0:0** ist die sog. *Unspecified Address*. Sie darf keinesfalls als Zieladresse in irgendwelchen IPv6-Paketen auftauchen.

Die *Unspecified Address* 0:0:0:0:0:0:0:0 kann jedoch als *Quell-Adresse* verwendet werden, bevor ein Host seine eigene Adresse gelernt hat. IPv6-Pakete mit dieser Adresse als Quelle werden selbstverständlich nicht geroutet.

Somit entspricht die *IPv6-Unspecified-Address* genau der Adresse 0.0.0.0 aus der IPv4-Welt.

Kapitel 3

IPv6 Adressen im DNS

Sicher werden Sie sich beim Experimentieren mit ping6 ärgern, wenn Sie jedesmal ellenlange IPv6-Adressen eintippen müssen. Also muss ein Nameserver her.

Es ist möglich, einem einzelnen FQDN (full qualified domain name) gleichzeitig mehrere IPv4 und IPv6 Adressen zuzuordnen.

Umgekehrt gilt auch, dass man einer Adresse mehrere DNS-Namen zuordnen kann. Das erlaubt, z.B. durch Einführen eine zusätzlichen Subdomäne, alle IPv6 Adressen in einer eigenen Zone zu verwalten. Die Zonendatei im Anhang B.3.5 zeigt beispielhaft solche DNS-Namen.

Für Versuche, aber auch für einen eventuellen produktiven Einsatz, empfiehlt es sich als DNS-Server BIND9 zu nehmen. [1].

Wie der zugehörige Dienst (named, gesprochen *name-dee*) konfiguriert wird, steht im Abschnitt B.

Wichtig ist an dieser Stelle nur zu beachten, welche *Resource-Record-Formate* und *Reverse-Pseudo-Domain-Namen* im Moment akutell und welche abgekündigt sind:

AAAA - Records - sind **aktuell**!

A6 - Records - sollten die AAAA-Records ersetzen, sind aber wieder in den Zustand "experimental" zurückgestuft worden.

Nibble Format - Dieses Format ist **aktuell** und wird in den *Reverse Zones* verwendet. Die Domain-Namen-Notation der Adresse **`2001:db8:fedc:abcd::1`** sieht so aus:

```
1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.d.c.b.a.c.d.e.f.8.b.d.0.1.0.0.2.ip6.arpa
```

Jeweils 4 bit (Nibble) der IPv6 Adresse werden als Hex-Ziffer notiert und bilden die Labels im ip6.arpa-Namensraum.

Bitlabel Format - Dieses Format ist **abgekündigt**. Die obige Adresse würde so aussehen:

```
\[20010db8fedcabcd0000000000000001/128].ip6.arpa
```

[1] Berklee Internet Name Domain / Daemon

Reverse Domain - **aktuell** ist:

```
ip6.arpa.
```

Reverse Domain - **abgekündigt** wurde:

```
ip6.int.
```

Kapitel 4

Übungen

Im folgenden sind einige einfache Versuche mit IPv6 beschrieben. Der Verkehr an der Netzwerkschnittstelle wurde dabei mit `tcpdump` aufgezeichnet.
Selbstverständlich geht das auch bequemer mit dem grafisch bedienbaren Programm `wireshark` [1].

4.0.1 Autoconfiguration und Duplicate Address Detection

4.0.1.1 Adresse ist nicht doppelt vorhanden

Auf einem System mit BSD-Kern wird mit einer Kommandofolge das Ethernet-Interface deaktiviert und anschliessend wieder aktiviert:

```
ifconfig en0 inet6 down
ifconfig en0 inet6 up
```

Das Ergebnis des ifconfig-Kommandos sieht anschliessend so aus:

```
en0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        inet6 fe80::21b:63ff:fe9f:a23c%en0 prefixlen 64 duplicated scopeid 0x4
        inet 192.168.1.105 netmask 0xffffff00 broadcast 192.168.1.255
        ether 00:1b:63:9f:a2:3c
             ...
```

Man erkennt, dass der Rechner sich selbst eine IP6-Adresse vergeben hat, die nach der Regel in Kap. 2.2.1 gebildet wurde.
In einer zweiten Konsole wird nun mit dem Kommando `tcpdump` der *Duplicate Adress Detection*-Vorgang beobachtet. Damit `tcpdump` startet, muss es vor dem Herunterfahren von `en0` aufgerufen werden:

```
tcpdump -nti en0 ip6
```

Das von `tcpdump` aufgezeichnete Ergebnis sieht so aus:

[1] Aber wer möchte sich schon dem Verdacht aussetzen, einen Computer nur mit der *Maus* bedienen zu können?

```
IP6 :: > ff02::1:ff9f:a23c: ICMP6, neighbor solicitation,
   who has fe80::21b:63ff:fe9f:a23c, length 24
```

Man erkennt als Quell-IP die unspezifizierte Adresse **::** . Die Zieladresse hat den Wert

`ff02::1:ff9f:a23c`

Diese Zieladresse wird nach folgender Regel gebildet:

1. nimm die 104-bit-lange Folge ff02::1:ff `(ff02:0000:0000:0000:0000:0001:ff)`
2. ergänze die 104-bit-Folge mit den letzten 24 bit der IPv6-Adresse (die natürlich auch den letzten 24 bit der MAC-Adresse entsprechen).

Erhält der Host auf seine Duplicate Address Detection - Nachricht keine Antwort nimmt er an, dass die Adresse verfügbar ist und verwendet diese.

4.0.1.2 Adresse ist bereits im Netz

Hier noch der von tcpdump aufgezeichnete Netzverkehr, wenn im Netzwerk bereits ein Host mit der Adresse `fe80::21b:63ff:fe9f:a23c` existiert.
Diese Adresse wurde dazu mit der Hand am Arm auf einem Linux-System mit dem Kommando

```
ifconfig eth0 inet6 add fe80::021b:63ff:fe9f:a23c
```

erzeugt.

```
IP6 fe80::21b:63ff:fe9f:a23c > ff02::1:ff9f:a23c: HBH ICMP6, multicast
   listener reportmax resp delay: 0 addr: ff02::1:ff9f:a23c, length 24

IP6 :: > ff02::1:ff9f:a23c: ICMP6, neighbor solicitation,
   who has fe80::21b:63ff:fe9f:a23c, length 24

IP6 fe80::21b:63ff:fe9f:a23c > ff02::1: ICMP6, neighbor advertisement,
   tgt is fe80::21b:63ff:fe9f:a23c, length 32

IP6 fe80::21b:63ff:fe9f:a23c > ff02::1:ff9f:a23c: HBH ICMP6, multicast
   listener reportmax resp delay: 0 addr: ff02::1:ff9f:a23c, length 24

IP6 :: > ff02::1:ff9f:a23c: ICMP6, neighbor solicitation,
   who has fe80::21b:63ff:fe9f:a23c, length 24

IP6 fe80::21b:63ff:fe9f:a23c > ff02::1: ICMP6, neighbor advertisement,
   tgt is fe80::21b:63ff:fe9f:a23c, length 32
```

Analysiert man den Netzverkehr, erkennt man, dass der Host, der die Adresse bereits verwendet auf die DAD-Nachricht antwortet. Dazu schickt er die ICMP6-Antwort an die **All-Nodes-Adresse `ff02::1`**, da der DAD-Sender ja noch nicht über eine Unicast-Adresse erreicht werden kann.

So lange der Adresskonflikt besteht, wird die doppelt vorkommende Adresse nicht verwendet, aber der initiierende Host prüft in Zeitabständen, ob der Konflikt noch besteht.

4.0.2 Neighbor Discovery

Neighbor Discovery (ND) dient unter IPv6 dazu, zu einer gegebenen IPv6-Zieladresse die zugehörige *physikalische Adresse* (MAC-Adresse) aufzulösen. Es ist somit ein Ersatz für das *Address Resolution Protocol* (ARP) von IPv4.

> Neighbor Discovery funktioniert sehr ähnlich wie die Duplicate Address Detection aus dem vorigen Abschnitt.
> Der einzige Unterschied: der Initiator verwendet nun nicht mehr die unspezifizierte Adresse ::, sondern seine Eigene.

Hier die ND-Sequenz, aufgezeichnet mit `tcpdump` . Um die Neighbor Discovery anzustossen, wurde mit `ndp -c` zunächst der ndp-Cache gelöscht. Anschliessend zwingt das Kommando

```
ping6 fe80::230:65ff:fe6a:971a%en0
```

den Host dazu, die Link-Layer-Adresse (MAC-Adr.) des Ziels aufzulösen:

```
IP6 fe80::21b:63ff:fe9f:a23c > ff02::1:ff6a:971a: ICMP6, neighbor
   solicitation, who has fe80::230:65ff:fe6a:971a, length 32

IP6 fe80::230:65ff:fe6a:971a > fe80::21b:63ff:fe9f:a23c: ICMP6,
   neighbor advertisement, tgt is fe80::230:65ff:fe6a:971a, length 32

IP6 fe80::21b:63ff:fe9f:a23c > fe80::230:65ff:fe6a:971a:
   ICMP6, echo request, seq 0, length 16
```

Die Ziel-Adresse des ersten Pakets wird wie in Kap. 4.0.1.1 beschrieben gebildet. Als Quelladresse wird nun natürlich die eigene IPv6-Adresse verwendet. Der verwendete ICMP6-Typ heisst **Neighbor Solicitation**.

Der Besitzer der angefragten IPv6-Adr. antwortet mit einer ICMP6-**Neighbor Advertisment** Nachricht.

Das dritte, aufgezeichnete Paket stammt vom schlussendlich ausgeführten ping6-Kommando.

4.0.3 Router Solicitation und Router Advertisment

In diesem Versuch soll das Ethernet-Interface eines Hosts, der sich im Netzwerk mit einem Router befindet, gestartet werden. Abb. 4.1 zeigt den Testaufbau mit IPv6-Adressen.

Die vollständige quagga-Konfiguration des Routers ist im Anhang abgedruckt (Anhang A).

Der Host sendet wieder eine ICMP6 *Neighbor Solicitation* Nachricht. Da die Adresse frei ist, bekommt er keine Antwort und er vergibt sich selbst diese Adresse.

Die nächste ICMP6-Nachricht nennt man *Router Soliciation*. Damit sucht der Host nach Routern im Netz.

Diese Nachrichten werden (zumindest vom Darwin-System des Autors) in Zeitabständen wiederholt, wenn kein Router antwortet.

Da in unserem Testaufbau ein Quagga-Router mit `ospf6d` läuft, antwortet dieser mit einer *Router Advertisment* Nachricht. Allerdings muss dazu vorab mit dem vtysh-Kommando

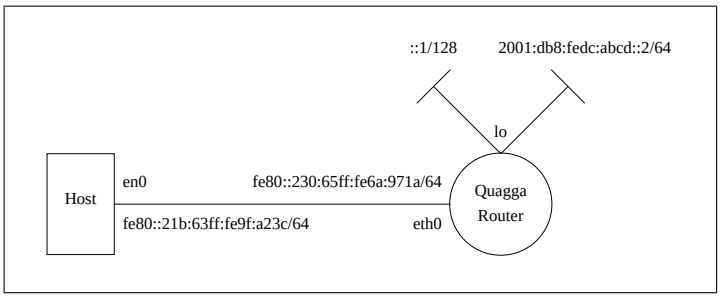


Abbildung 4.1: Testaufbau für das Router Advertisement

```
(config-if)# no ipv6 nd suppress-ra
```

das Router-Advertisement auf dem Interface eth0 erlaubt werden.

Zusätzlich zu dieser *erbetenen* Antwort senden Router auch regelmässig von sich aus sog. *Unsolicited Router Advertisments*. Dies erfolgt in zufälligen Zeitabständen von 200 bis 600 Sekunden.

```
IP6 :: > ff02::1:ff9f:a23c: ICMP6, neighbor solicitation,
   who has fe80::21b:63ff:fe9f:a23c, length 24

IP6 fe80::21b:63ff:fe9f:a23c > ff02::2: ICMP6,
   router solicitation, length 16

IP6 fe80::230:65ff:fe6a:971a > ff02::1: ICMP6,
   router advertisement, length 24
```

Nachdem der Router (`fe80::230:65ff:fe6a:971a`) sich mit dem Router Advertisment bekannt gemacht hat, taucht er auch in der Routingtabelle des Hosts auf:

```
sh-3.2# netstat -rn
Routing tables

                     ...

Internet6:
Destination                             Gateway                         Flags      Netif Expire
default                                 fe80::230:65ff:fe6a:971a%en0    UGc         en0
::1                                     link#1                          UHL         lo0
fe80::%lo0/64                           fe80::1%lo0                     Uc          lo0
fe80::1%lo0                             link#1                          UHL         lo0
fe80::%en0/64                           link#4                          UC          en0
fe80::21b:63ff:fe9f:a23c%en0            0:1b:63:9f:a2:3c                UHL         lo0
fe80::230:65ff:fe6a:971a%en0            0:30:65:6a:97:1a                UHLW        en0
fe80::%en1/64                           link#6                          UC          en1
fe80::21c:b3ff:fec3:830f%en1            0:1c:b3:c3:83:f                 UHL         lo0
ff01::/32                               ::1                             U           lo0
ff02::/32                               fe80::1%lo0                     UC          lo0
ff02::/32                               link#4                          UC          en0
ff02::/32                               link#6                          UC          en1
```

Nun können wir versuchen, mit ping6 das Loopback-Interface des Routers zu erreichen:

```
sh-3.2# ping6 2001:db8:fedc:abcd::2%en0
PING6(56=40+8+8 bytes) fe80::21b:63ff:fe9f:a23c%en0 --> 2001:db8:fedc:abcd::2
16 bytes from 2001:db8:fedc:abcd::2, icmp_seq=0 hlim=64 time=3.786 ms
16 bytes from 2001:db8:fedc:abcd::2, icmp_seq=1 hlim=64 time=0.463 ms
16 bytes from 2001:db8:fedc:abcd::2, icmp_seq=2 hlim=64 time=0.452 ms
       ...
```

Anhang A

Quagga-Konfiguration für Router-Advertisment

A.1 quagga - Schnellanleitung

Hier eine sehr kurze Anleitung zur Inbetriebnahme von `quagga` mit dem `ospf6d`.

Installation - Auf Systemen die `aptitude` unterstützen kann man so installieren (die Doku ist natürlich optional):

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

Routingprotokolle auswählen - Um das gewünschte Routingprotokoll zu starten, muss man die Datei

```
/etc/quagga/daemons
```

editieren. Der AbschnittA.2.1 zeigt ein Beispiel für **ospf6**.

Virtuelle Konsolen - Die Dienste von *quagga* lassen sich per `telnet` auf dem lokalen Host **einzeln** über folgende Ports erreichen:

zebra	2601
ripd	2602
ripng	2603
ospfd	2604
bgpd	2605
ospf6d	2606

Wesentlich komfortabler ist es, mit dem Kommando `vtysh` zu arbeiten, da hat man alles unter **einer** virtuellen Konsole. Um `vtysh` zu aktivieren, muss lediglich die Konfigurationsatei

```
/etc/quagga/vtysh.conf
```

erzeugt werden. Ein Beispiel ist wiederum im Abschnitt A.2.2 abgedruckt.

Pager setzen - Normalerweise ist die Umgebungsvariable für den Pager auf `less` gesetzt. In

```
/etc/environment
```

sollte deshalb

```
VTYSH_PAGER=more
```

eingetragen werden, damit der Pager auch ohne Betätigen der Taste 'Q' wieder beendet wird.

Neustart - Neustart von `quagga`:

```
/etc/init.d/quagga restart
```

Virtuelle Konsole starten - um ins hassgeliebte IOS der virtuellen Konsole zu gelangen startet man diese mit:

```
vtysh
```

Routerkonfiguration sichern - mit dem IOS-Kommando `write` können Sie von innerhalb der *vtysh* Ihre Konfiguration sichern. Es wird dabei mit der unter A.2.2 angegebenen Datei für jeden aktivierten Dienst eine eigene Konfig-Datei erzeugt.

Abschnitte A.2.3 und A.2.4 zeigen Beispiele.

A.2 Konfigurationsdateien

A.2.1 daemons

```
# This file tells the quagga package which daemons to start.
#      ========== vom autor gekuerzt ============
#
zebra=yes
bgpd=no
ospfd=no
ospf6d=yes
ripd=no
ripngd=no
isisd=no
```

A.2.2 vtysh.conf

```
!
! Sample configuration file for vtysh.
!
!service integrated-vtysh-config
hostname routerXubbie
username root nopassword
!
```

A.2.3 zebra.conf

```
!
! Zebra configuration saved from vty
!   2009/06/19 22:24:46
!
hostname Router
password zebra
enable password zebra
!
interface eth0
 no ipv6 nd suppress-ra
!
interface lo
 ipv6 address 2001:db8:fedc:abcd::2/64
!
interface sit0
 ipv6 nd suppress-ra
!
ipv6 forwarding
!
!
line vty
!
```

A.2.4 ospf6d.conf

```
!
! Zebra configuration saved from vty
!   2009/06/19 22:24:46
!
hostname ospf6d@xubbie
password zebra
log stdout
service advanced-vty
!
debug ospf6 lsa unknown
debug ospf6 neighbor state
!
interface fxp0
 ipv6 ospf6 cost 1
 ipv6 ospf6 hello-interval 10
 ipv6 ospf6 dead-interval 40
 ipv6 ospf6 retransmit-interval 5
 ipv6 ospf6 priority 0
 ipv6 ospf6 transmit-delay 1
 ipv6 ospf6 instance-id 0
!
interface lo
 ipv6 ospf6 cost 1
 ipv6 ospf6 hello-interval 10
 ipv6 ospf6 dead-interval 40
 ipv6 ospf6 retransmit-interval 5
 ipv6 ospf6 priority 1
 ipv6 ospf6 transmit-delay 1
 ipv6 ospf6 instance-id 0
!
interface lo0
 ipv6 ospf6 cost 1
 ipv6 ospf6 hello-interval 10
 ipv6 ospf6 dead-interval 40
 ipv6 ospf6 retransmit-interval 5
 ipv6 ospf6 priority 1
 ipv6 ospf6 transmit-delay 1
 ipv6 ospf6 instance-id 0
!
router ospf6
 router-id 255.1.1.1
 redistribute static
 interface fxp0 area 0.0.0.0
 interface lo area 0.0.0.0
!
access-list access4 permit 127.0.0.1/32
!
ipv6 access-list access6 permit 3ffe:501::/32
ipv6 access-list access6 permit 2001:200::/48
ipv6 access-list access6 permit ::1/128
!
ipv6 prefix-list test-prefix seq 1000 deny any
!
route-map static-ospf6 permit 10
 match ipv6 address prefix-list test-prefix
 set metric-type type-2
 set metric 2000
!
line vty
 access-class access4
 ipv6 access-class access6
 exec-timeout 0 0
!
```

Anhang B

Konfiguration von Bind9

Folgender Text beschreibt nur rudimentär die Konfiguration von Bind9 auf Ubunut-basierten Systemen (konkret: xubuntu8.04). Da Bind9 aber **der** Internetstandard überhaupt ist, ist der Dienst auf fast allen Systemen mit Unix-Kern zu finden.

Je nach System ist lediglich der *Speicherort* der Konfigurations- und Zonendateien verschieden. Beispiele:

ubuntu - Hier werden Konfigurations- und Zonendateien gemeinsam im Verzeichnis

```
/etc/bind/
```

gespeichert. Das ist aber willkürlich, da die Pfade zu den Zonendateien in der zentralen Konfigurationsdatei `named.conf` als *absolute Pfade* angegeben werden.

darwin-osx - osx speichert die Zonendateien in `/var/named` . Die Datei `/etc/named.conf` enthält Pfade **relativ** zu `/var/named` .

```
Konfiguration: /etc/named.conf
Zonendateien:  /var/named/
```

B.1 Testaufbau

Abb. B.1 zeigt eine minimale Topologie mit den verwendeten IPv6-Adressen.

B.2 Internet-Anbindung und Fake-Root-Zone

Falls die Testumgebung keine Anbindung ans Internet hat, wird der Namerserver die Root-Server nicht finden und erst nach einer Wartezeit eine negative Antwort erzeugen. Bei Namen, die er nicht selbst auflösen kann (die also nicht zu seiner Zone gehören) hat muss man deshalb eine Verzögerung in Kauf nehmen.

Man kann jedoch eine Fake-Root-Zone erzeugen, dann kommt die negative Antwort verzögerunsfrei. In der in B.3.1 abgedruckten Datei ist die Verwendung einer Fake-

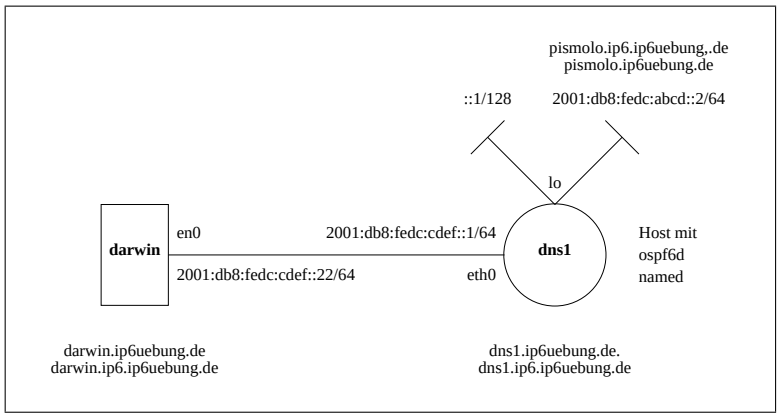


Abbildung B.1: Testaufbau für Bind9 mit IPv6

Root-Zone enthalten, aber auskommentiert. Die Fake-Root-Zonendatei ist hier B.3.4 abgedruckt.

B.3 Konfigurationsdateien

B.3.1 named.conf

```
// This is the primary configuration file for the BIND DNS server named.
//
// Please read /usr/share/doc/bind9/README.Debian.gz for information on the
// structure of BIND configuration files in Debian, *BEFORE* you customize
// this configuration file.
//
// If you are just adding zones, please do that in /etc/bind/named.conf.local

include "/etc/bind/named.conf.options";

// prime the server with knowledge of the root servers
//auskommentieren um fake-root-zone verwenden zu koennen

zone "." {
        type hint;
        file "/etc/bind/db.root";
};

//fake-root-zone fuer ip6uebung
//zone "." {
//        type master;
//        file "/etc/bind/db.fakeroot";
//};

// be authoritative for the localhost forward and reverse zones, and for
// broadcast zones as per RFC 1912

zone "localhost" {
        type master;
        file "/etc/bind/db.local";
};

zone "127.in-addr.arpa" {
        type master;
        file "/etc/bind/db.127";
};

zone "0.in-addr.arpa" {
        type master;
        file "/etc/bind/db.0";
};

zone "255.in-addr.arpa" {
        type master;
        file "/etc/bind/db.255";
};

// zone "com" { type delegation-only; };
// zone "net" { type delegation-only; };

// From the release notes:
//  Because many of our users are uncomfortable receiving undelegated answers
//  from root or top level domains, other than a few for whom that behaviour
//  has been trusted and expected for quite some length of time, we have now
//  introduced the "root-delegations-only" feature which applies delegation-only
//  logic to all top level domains, and to the root domain.  An exception list
//  should be specified, including "MUSEUM" and "DE", and any other top level
//  domains from whom undelegated responses are expected and trusted.
// root-delegation-only exclude { "DE"; "MUSEUM"; };

include "/etc/bind/named.conf.local";

include "/etc/bind/named.conf.ip6uebung";
```

B.3.2 named.conf.options

```
options {
        directory "/var/cache/bind";

        // If there is a firewall between you and nameservers you want
        // to talk to, you might need to uncomment the query-source
        // directive below.  Previous versions of BIND always asked
        // questions using port 53, but BIND 8.1 and later use an unprivileged
        // port by default.

        // query-source address * port 53;

        // If your ISP provided one or more IP addresses for stable
        // nameservers, you probably want to use them as forwarders.
        // Uncomment the following block, and insert the addresses replacing
        // the all-0's placeholder.

        // forwarders {
        //         0.0.0.0;
        // };

        auth-nxdomain no;    # conform to RFC1035
        listen-on-v6 { any; };
        allow-query { any; };

        forwarders{
                192.168.2.1;
        };
};
```

B.3.3 named.conf.ip6uebung

```
zone "ip6uebung.de"{
        type master;
        file "/etc/bind/db.ip6uebung";
};

zone "c.d.e.f.8.b.d.0.1.0.0.2.ip6.arpa."{
        type master;
        file "/etc/bind/db.2001.db8.fedc";
};
```

B.3.4 db.fakeroot

```
$TTL 3600
@ IN        SOA        dns1.ip6uebung.de. root.dns1.ip6uebung.de. (
                        5         ;seriennummer
                        15m         ;refresh
                        5m         ;retry
                        30d         ;expire
                        1h)        ;ttl negative cache
;
@        IN        NS        dns1.ip6uebung.de.

localhost        A        127.0.0.1
                AAAA        ::1
```

B.3.5 db.ip6uebung

```
$TTL 3600
@ IN        SOA        dns1.ip6uebung.de. root.dns1.ip6uebung.de. (
                        5         ;seriennummer
                        15m         ;refresh
                        5m         ;retry
                        30d         ;expire
                        1h)        ;ttl negative cache
;
@        IN        NS        dns1.ip6uebung.de.

localhost        A        127.0.0.1
                AAAA        ::1

dns1                AAAA        2001:db8:fedc:cdef::1
dns1.ip6        AAAA        2001:db8:fedc:cdef::1

darwin                AAAA        2001:db8:fedc:cdef::22
darwin.ip6        AAAA        2001:db8:fedc:cdef::22
mbpro15                AAAA        2001:db8:fedc:cdef::22
mbpro15.ip6        AAAA        2001:db8:fedc:cdef::22
pismolo                AAAA        2001:db8:fedc:abcd::2
pismolo.ip6        AAAA        2001:db8:fedc:abcd::2
```

B.3.6 db.2001.db8.fedc

```
$TTL 3600
@ IN        SOA        dns1.ip6uebung.de. root.dns1.ip6uebung.de. (
                        5         ;seriennummer
                        15m         ;refresh
                        5m         ;retry
                        30d         ;expire
                        1h)        ;ttl negative cache
;
@        IN        NS        dns1.ip6uebung.de.

;die domain ip6.in. ist abgekuendigt
;im beispiel verwenden wir das nibble-format

; die naechste kommentarzeile dient als lineal
;             1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6
$ORIGIN        f.e.d.c.c.d.e.f.8.b.d.0.1.0.0.2.ip6.arpa.

; 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6
1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0        PTR        dns1.ip6uebung.de.
1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0        PTR        dns1.ip6.ip6uebung.de.
2.2.0.0.0.0.0.0.0.0.0.0.0.0.0.0        PTR        darwin.ip6uebung.de.
2.2.0.0.0.0.0.0.0.0.0.0.0.0.0.0        PTR        darwin.ip6.ip6uebung.de.
2.2.0.0.0.0.0.0.0.0.0.0.0.0.0.0        PTR        mbpro15.ip6uebung.de.
2.2.0.0.0.0.0.0.0.0.0.0.0.0.0.0        PTR        mbpro15.ip6.ip6uebung.de.

;             1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6
$ORIGIN        d.c.b.a.c.d.e.f.8.b.d.0.1.0.0.2.ip6.arpa.
2.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0 PTR     pismolo.ip6uebung.de.
2.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0 PTR     pismolo.ip6.ip6uebung.de.
```

B.4 Bind9/named unter OSX1.5

```
launchctl unload org.isc.named.plist
launchctl load /System/Library/LaunchDaemons/org.isc.named.plist
launchctl stop org.isc.named
launchctl start org.isc.named

rndc-confgen

rndc reload
```

B.4.1 org.isc.named.plist

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN"
  "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
        <key>Disabled</key>
        <false/>
        <key>Label</key>
        <string>org.isc.named</string>
        <key>OnDemand</key>
        <false/>
        <key>ProgramArguments</key>
        <array>
<!--
        Un-comment the following lines to run named with a sandbox profile.
        Sandbox profiles restrict processes from performing unauthorized
        operations; so it may be necessary to update the profile
        (/usr/share/sandbox/named.sb) if any changes are made to the named
        configuration (/etc/named.conf).
-->
<!--
                <string>/usr/bin/sandbox-exec</string>
                <string>-f</string>
                <string>/usr/share/sandbox/named.sb</string>
-->
                <string>/usr/sbin/named</string>
                <string>-f</string>
        </array>
        <key>ServiceIPC</key>
        <false/>
</dict>
</plist>
```

B.4.2 /etc/rndc.conf

```
# Start of rndc.conf
key "rndc-key" {
        algorithm hmac-md5;
        secret "BgeHM04C3vsGlE6BHW/tuA==";
};

options {
        default-key "rndc-key";
        default-server 127.0.0.1;
        default-port 953;
};
# End of rndc.conf

# Use with the following in named.conf, adjusting the allow list as needed:
# key "rndc-key" {
#         algorithm hmac-md5;
#         secret "BgeHM04C3vsGlE6BHW/tuA==";
# };
#
# controls {
#         inet 127.0.0.1 port 953
#                 allow { 127.0.0.1; } keys { "rndc-key"; };
# };
# End of named.conf
```

B.4.3 /etc/named.conf

```
//
// Default controls
//

# Use with the following in named.conf, adjusting the allow list as needed:
 key "rndc-key" {
         algorithm hmac-md5;
         secret "BgeHM04C3vsGlE6BHW/tuA==";
 };

controls {
        inet 127.0.0.1 port 54 allow {any;} keys { "rndc-key"; };
};

options {
        directory "/var/named";
        auth-nxdomain no;    # conform to RFC1035
        listen-on-v6 { any; };
        allow-query { any; };
};
//
// a caching only nameserver config
//
zone "." IN {
        type hint;
        file "named.ca";
};

zone "localhost" IN {
        type master;
        file "localhost.zone";
        allow-update { none; };
};

zone "0.0.127.in-addr.arpa" IN {
        type master;
        file "named.local";
        allow-update { none; };
};

zone "ip6uebung.de"{
        type master;
        file "/var/named/db.ip6uebung";
};

zone "c.d.e.f.8.b.d.0.1.0.0.2.ip6.arpa."{
        type master;
        file "/var/named/db.2001.db8.fedc";
};


logging {
        category default {
                _default_log;
        };

        channel _default_log  {
                file "/Library/Logs/named.log";
                severity info;
                print-time yes;
        };
};
```

Anhang C

Statische Routen unter BSD und GNU/Linux

Auf einem Mac lässt sich eine statische Router so eintragen:

```
route add -inet6 2001:db8:fedc:abcd:: 2001:db8:fedc:cdef::1
route add -inet6 net 2001:db8:fedc:abcd:: 2001:db8:fedc:cdef::1
```

Beide Kommandos scheinen gleichwertig zu sein. Wenn man einen Präfix wie z.B. /64 angibt, erhält man die Meldung “bad value”:

```
route add -inet6 2001:db8:fedc:abcd::/64 2001:db8:fedc:cdef::1
2001:db8:fedc:abcd::/64: bad value
```

Unter GNU/Linux sieht das Kommando so aus:

```
route --inet6 add  2001:db8:fedc:abcd:: gw 2001:db8:fedc:cdef::1
2001:db8:fedc:abcd::/64: bad value
```

Anhang D

Wichtige Header-Werte

D.1 Next Header Feldwerte

Die von der IANA vergebenen Protokoll-Nummern sind hier dokumentiert:

```
http://www.iana.org/assignments/protocol-numbers/
```

Die wichtigsten Werte finden Sie in folgender Tabelle:

Value (Hex)	Value (Dez)	Protocol / Extension Header
00	0	Hop-By-Hop Options Extension Header
01	1	ICMPv4
02	2	IGMPv4
04	4	IP in IP Encapsulation
06	6	TCP
08	8	EGP
11	17	UDP
29	41	IPv6
2B	43	Routing Extension Header
2C	44	Fragmentation Extension Header
2E	46	Resource Reservation Protocol (RSVP)
32	50	Encrypted Security Payload (ESP) Extension Header
33	51	Authentication Header (AH) Extension Header
3A	58	ICMPv6
3B	59	No Next Header
3C	60	Destination Options Extension Header

Tabelle D.1: Wichtige Next-Header-Werte

D.2 Werte des Scope-Nibble

Wert	Bedeutung
0	reserved
1	Interface-Local scope
2	Link-Local scope
3	reserved
4	Admin-Local scope
5	Site-Local scope
6	(unassigned)
7	(unassigned)
8	Organization-Local scope
9	(unassigned)
A	(unassigned)
B	(unassigned)
C	(unassigned)
D	(unassigned)
E	Global scope
F	reserved

Tabelle D.2: Scope-Values

D.3 IPv6 Kommandos unter Linux

Die folgenden Befehle funktionieren auf jeden Fall unter Linux. Auf einem BSD-System (z.B. Mac-OSX) gelten z.T. andere Optionen und es gibt das ip-Kommando nicht.

Unter OSX kann man Adressen mit ifconfig vergeben, es wird jedoch empfohlen, stattdessen das Kommando /usr/sbin/networksetup zu verwenden (vgl. D.4).

D.3.1 ping6

D.3.1.1 ping6 auf eine Link-Lokale Unicast-Adresse

Da alle link-lokalen Adressen auf allen Interfaces eines Rechners zum Subnetz fe80::/10 gehören, kann der Kernel nicht herausfinden, über welches Interface das ping6-Kommando gesendet werden soll.

Das Interface muss in diesem Fall explizit mitangegeben werden!

```
ping6 fe80::230:65ff:fe6a:971a%en0
```

D.3.1.2 ping6 auf eine globale Unicast-Adresse

Hier genügt die Angabe der Adresse. Das Interface findet der Kernel über seine Routingtabelle. Gibt es einen DNS-Eintrag, genügt natürlich auch der DNS-Name:

```
ping6 2001:7c0:e100:1::1
ping6 www.heise.de
```

D.3.2 traceroute6 und tracepath6

`tracepath` ist dem bekannten `traceroute` sehr ähnlich, nur beötigt es keine Superuser-Rechte. Das Gleiche gilt auch für die IPv6-Versionen:

```
traceroute6 www.heise.de
tracepath6 www.heise.de
```

D.3.3 Adresse auf Interface konfigurieren

Unter Linux gibt es das IP-Kommando Paket von Alexey N. Kuznetsov. Da dieser Befehl wesentlich mächtiger ist als das alte ifconfig, wir an dieser Stelle nur auf das ip-Kommando eingegangen.

Hier ein paar Beispiele, die alle genau das gleiche bewirken:

```
ip addr add 2001:7c0:e100:1::1/64 dev eth0
ip -familiy inet6 addr add 2001:7c0:e100:1::1/64 dev eth0
ip -f inet6 addr add 2001:7c0:e100:1::1/64 dev eth0
ip -6 addr add 2001:7c0:e100:1::1/64 dev eth0
```

Entfernen mit `del` statt `add`:

```
ip -6 addr del 2001:7c0:e100:1::1/64 dev eth0
```

D.3.4 IPv6-Adressen eines Interfaces auflisten lassen

Um sich nur die IPv6-Adressen eines Interfaces anzeigen zu lassen, kann man folgenden Befehl verwenden:

```
ip -6 addr show eth0
```

D.3.5 Neighbor-Discovery-Cache auflisten

Linux:

```
ip -6 neigh show
```

BSD:

```
ndp -a
```

D.3.6 Eine Route eintragen

```
ip -6 route add 2001:7c0:e100:aaaa::/64 via 2001:7c0:e100:1::ffff
```

Auch hier gilt wiederum: wird `add` durch `del` ersetzt, wird die Route aus der Kerneltabelle wieder entfernt.

D.4 Das networksetup-Kommando unter OSX

Networkservices auflisten lassen:

```
networksetup -listallnetworkservices
```

Konfiguration eines Networkservices anzeigen lassen

```
networksetup -getinfo ethernet
```