Linux IPv6 HOWTO (de)

Peter Bieringer

pb at bieringer dot de

Versionsgeschichte

Version 0.63.de.1	2009-02-14	Geändert durch: PB
Version 0.62.de.1	2008-11-09	Geändert durch: PB
Version 0.61.1.de.2	2008-05-17	Geändert durch: PB
Version 0.61.1.de.1	2007-11-11	Geändert durch: PB
Version 0.61.de.1	2007-10-06	Geändert durch: PB

Das Ziel des Linux IPv6 HOWTO ist die Beantwortung von Basis- und Experten-Fragen zum Thema IPv6 mit Linux-Betriebssystemen. Dieses HOWTO will dem Leser genug Informationen bereitstellen, um IPv6-Anwendungen auf Linux Computer installieren, konfigurieren und anwenden zu können. Zwischen-Versionen sind auf mirrors.bieringer.de oder mirrors.deepspace6.net verfügbar. Änderungen sind in der Revisionshistorie aufgelistet.

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Kapitel 1. Allgemein

Informationen über verfügbare Übersetzungen finden Sie im Abschnitt Übersetzungen.

1.1. Copyright, Lizenz und anderes

1.1.1. Copyright

1.1.2. Lizenz

1.1.3. Über den Autor

1.1.3.1. Internet/IPv6 Background des Autors

- 1993: In Kontakt mit dem Internet getreten, kennenlernen von konsolenbasierte E-Mail- und News-Client-Programme (z.B.: suchen Sie nach "e91abier" auf groups.google.com, das ist der Autor).
- 1996: Anfrage zur Gestaltung eines IPv6 Kurses inklusive eines Workshops zum Thema Linux Betriebssystem bekommen.
- 1997: Schreiben einer Anleitung, wie man IPv6 auf Linux Systemen installieren, konfigurieren und anwenden kann, genannt IPv6 & Linux - HowTo (siehe unter IPv6 & Linux - HowTo/History).
- 2001: Begonnen, dieses neue Linux IPv6 HOWTO zu schreiben.

1.1.3.2. Ansprechpartner

Der Autor kann via E-Mail <pb at bieringer dot de> und auch über seine homepage kontaktiert werden.

Der Autor lebt zurzeit in München (nördlicher Teil von Schwabing) / Bayern / (Süd-)Deutschland / (Mittel-)Europa / Erde (Oberfläche / Festland).

1.1.3.3. Ansprechpartner für Übersetzungen

Der Autor der deutschsprachigen Version kann via E-Mail unter <gkaefer at gmx dot at> kontaktiert werden.

1.2. Kategorie

Dieses HOWTO sollte in der Kategorie "*Networking/Protocols*" aufgelistet werden.

1.3. Version, Werdegang und Unerledigtes

1.3.1. Version

Die aktuelle Versionsnummer finden Sie auf der Titelseite.

CVS-Information: CVS-ID: $Id: Linux+IPv6-HOWTO.de.lyx,v 1.56 2006/11/08 22:45:26 pbldp Exp $

Für andere verfügbare Versionen/Übersetzungen siehe auch http://www.bieringer.de/linux/IPv6/.

1.3.2. Werdegang

1.3.2.1. Eckpunkte

2001-11-30: Beginn mit der Neukonzeption dieses HOWTOs.

2002-01-02: Viel Inhalt eingearbeitet, erste Version des Kapitels 1 veröffentlicht (Version 0.10).

2002-01-14: Weitere Vervollständigung und Überprüfung des Inhalts, öffentliche Freigabe des kompletten Dokuments (Version 0.14).

2002-08-16: Polnische Übersetzung ist in Arbeit.

2002-10-31: Chinesische Übersetzung ist verfügbar.

2002-11-10: Deutsche Übersetzung ist in Arbeit.

2003-02-10: Deutsche Übersetzung ist verfügbar.

2003-04-09: Französische Übersetzung ist in Arbeit.

2003-05-09: Französische Übersetzung ist verfügbar

2003-10-16: Italienische Übersetzung ist in Arbeit.

2004-03-12: Italienische Übersetzung ist verfügbar

2005-07-25: Türkische Übersetzung ist verfügbar

2008-07-30: Spanische Übersetzung ist verfügbar (aber noch nicht abgeschlossen)

1.3.2.2. Vollständiger Werdegang

Am Ende dieses Dokumentes finden Sie die Historie der Änderungen.

1.3.3. Unerledigtes

- Fehlenden Inhalt ergänzen
- Grammatik-Überprüfung beenden

1.4. Übersetzungen

Übersetzungen müssen den URL, die Versionsnummer und das Copyright des Originaldokuments enthalten (aber auch die Daten zu ihrer Übersetzung). Bitte übersetzen Sie nicht das Original-Changelog, das wäre nicht sehr sinnvoll. Es sieht aus, als wäre die Änderungsfrequenz dieses Dokumentes überwiegend geringer als einmal pro Monat. Seit Version 0.27 scheint der Hauptteil des vom Autor beigesteuerten Inhaltes geschrieben zu sein. Übersetzungen müssen immer das englische Original als Quelle benutzen.

1.4.1. Diverse Sprachen

Notiz: eine Übersicht mit URLs ist zu finden unter http://www.bieringer.de/linux/IPv6/.

1.4.1.1. Deutsch

Mit 2002-11-10 wurde von Georg Käfer <gkaefer at gmx dot at> die Deutsche Übersetzung begonnen und am 10.02.2003 erstmals publiziert. Grundlage der Übersetzung ist die CVS-Version 1.53 der Lyx-Datei, aus der das Linux IPv6 HOWTO Version 0.39.2 erstellt wurde.

Mit Sicherheit werden in dieser Übersetzung noch so manche 'Hoppalas' und auch 'grausige' Rechtschreibfehler zu finden sein. Sorry! Ebenfalls fällt es nicht immer leicht, zu entscheiden, ob ein Terminus überhaupt übersetzt werden soll, bzw. ob die Beibehaltung des englischen Originalwortes nicht zielführender wäre...

Neue deutsche Rechtschreibung: es wurde eine konservative Übersetzung angewandt, d.h. nicht alle möglichen Änderungen der Neuen deutschen Rechtschreibung wurden durchgeführt, sondern nur die notwendigen Änderungen.

Die URL für diese deutsche Übersetzung ist: http://mirrors.deepspace6.net/Linux+IPv6-HOWTO-de/

1.4.1.2. Andere Sprachen

Informationen über Übersetzungen in andere Sprachen finden Sie hier im Originaldokument: TLDP / Linux+IPv6-HOWTO / Übersetzungen.

1.5. Technisches

1.5.1. Originalquelle dieses HOWTOs

Die originale englische Version dieses HOWTOs wurde mit LyX Version 1.6.1 auf einem Fedora 10 Linux System mit SGML-Template (DocBook book) erstellt. Alle Dateien sind unter TLDP-CVS / users / Peter-Bieringer verfügbar.

Auch die deutsche Version wurde mit LyX erstellt und befindet sich ebenfalls im angegebenen CVS-Verzeichnis.

1.5.1.1. Zeilenumbruch in Code-Beispielen

Der Zeilenumbruch wird mit Hilfe eines selbst geschriebenen Tools "lyxcodelinewrapper.pl" erstellt; Sie finden das Skript am CVS unter: TLDP-CVS / users / Peter-Bieringer.

1.5.1.2. SGML Erzeugung

SGML wird mit Hilfe der Exportfunktion in LyX generiert.

Um korrekten SGML Code zu erstellen, müssen einige Korrekturen gemacht werden. Die entsprechenden Perl Skripts finden Sie unter TLDP-CVS / users / Peter-Bieringer:

- Der Export von Lyx-Tabellen erstellt keine korrekten "colspan" Tags. Tool für die Korrektur: "sgmllyxtabletagfix.pl" (behoben seit LyX Version 1.2.0)
- LyX verwendet manchmal , anstelle der normalen, spezielle Links-/Rechts-Formatierungen für Zitate, die dann auch im generierten HTML Code ausgegeben werden. Einige Browser können das Ergebnis nicht besonders schön darstellen (bekannt sind: Opera TP2 oder Konqueror). Tool für die Korrektur: "sgmllyxquotefix.pl"

1.5.2. Online-Verweise auf die HTML Version dieses HOWTOs (Links / Anchors)

1.5.2.1. Hauptindexseite

Im Allgemeinen wird ein Verweis auf die Hauptindexseite empfohlen.

1.5.2.2. Seitennamen

Da die HTML-Seiten aus einer SGML-Datei erstellt werden, werden einige HTML-Dateinamen ziemlich zufällig gewählt. Manche Seiten sind jedoch in LyX mit Tags gekennzeichnet, woraus statische Namen resultieren. Diese sollten der besseren Referenz wegen zukünftig nicht geändert werden.

Bitte lassen Sie es wissen, wenn Sie glauben, dass ein Tag vergessen wurde.

1.6. Vorwort

Einiges vorab:

1.6.1. Wie viele IPv6 & Linux bezogene HOWTOs gibt es?

Inklusive diesem gibt es drei (3) HOWTO-Dokumente. Pardon, wenn das zu viele sind ;-)

1.6.1.1. Linux Ipv6 FAQ/HOWTO (veraltet)

Das erste IPv6 bezogene Dokument wurde von *Eric Osborne* geschrieben und heißt Linux IPv6 FAQ/HOWTO (bitte benutzen Sie den Text nur im historischen Kontext). Die neueste Version 3.2.1 wurde am 14.Juli 1997 veröffentlicht.

Bitte um Ihre Mithilfe: Wenn jemand das Erstellungsdatum der Erstversion dieses HOWTOs kennen sollte, senden Sie mir Bitte ein E-Mail (Die Information wird im Abschnitt "Werdegang" eingearbeitet).

1.6.1.2. IPv6 & Linux - HowTo (gewartet)

Ein zweites Dokument (IPv6 & Linux - HowTo) wurde vom selben Autor (Peter Bieringer) geschrieben und liegt im HTML-Format vor. Begonnen wurde mit dem Schreiben im April 1997 und die erste englische Version wurde im Juni 1997 veröffentlicht. Das Dokument wird weiterhin betreut, es wird aber langsam (jedoch nicht komplett) in das Linux IPv6 HOWTO - das Sie gerade lesen - eingearbeitet.

1.6.1.3. Linux IPv6 HOWTO (dieses Dokument)

Da das IPv6 & Linux - HowTo in HTML geschrieben wurde, war es nicht wirklich mit dem The Linux Documentation Project (TLDP) kompatibel. Der Autor (Peter Bieringer) bekam Ende Nov. 2001 die Anfrage, das IPv6 & Linux - HowTo in SGML zu konvertieren. Er entschied sich auf Grund dieser Diskontinuität (Future of IPv6 & Linux - HowTo) und der Tatsache, dass IPv6 mehr und mehr zum Standard wird, zum Schreiben eines neuen Dokuments. Im zweiten HOWTO (IPv6 & Linux - HowTo) wird auch weiterhin dynamischer Inhalt sowie weiterführender Inhalt zu finden sein.

1.7. Verwendete Begriffe, Glossar und Abkürzungen

1.7.1. Netzwerkbegriffe

Base 10
: Dezimales Zahlensystem, das die Zahlen 0-9 beinhaltet.

Base 16
: Generell in Programmiersprachen verwendetes hexedezimales Zahlensystem, das die Zahlen 0-9 und die Buchstaben A-F beinhaltet (Groß/Kleinschreibung möglich).

Base 85
: 85 verschiedene Zahlen/Buchstaben umfasst dieses Zahlensystem und ermöglicht dadurch kürzere Zeichenketten - aber niemals in der Praxis gesehen.

Bit
: Kleinste Speichereinheit mit dem Wert ein/wahr (1) oder aus/falsch (0)

Byte
: Meistens eine Menge von 8 bits (aber kein Muss - siehe ältere Computer Systeme)

Device
: Netzwerkgerät, siehe auch NIC

Dual homed host
: Ein Dual homed host ist ein Node mit zwei (physischen oder virtuellen) Schnittstellen auf zwei unterschiedlichen Links. Datenpakete können zwischen den zwei Verbindungen nicht weitergeleitet

werden.

Host
: Im Regelfall handelt es sich um einen Rechner mit einen Link sowie einer aktiven Netzwerk-Schnittstelle, z.B. Ethernet oder (aber nicht und) PPP.

Interface
: Ident mit "device", siehe auch NIC.

IP Header
: Kopf eines IP-Paketes (jedes Netzwerk-Paket hat einen header, die Form des headers ist abhängig von der Netzwerkschicht).

Link
: Ein Link ist eine Schicht 2 Netzwerk-Transportmedium für Pakete; Beispiele sind Ethernet, Token Ring, PPP, SLIP, ATM, ISDN, Frame Relay,...

Node
: Ein Node (=Knoten) ist ein Host oder ein Router.

Octet
: Sammlung von acht (8) realen bits, vergleichbar mit "byte".

Port
: Information für den TCP/UDP dispatcher (Schicht 4), mit dessen Hilfe Informationen auf höhere Schichten transportiert werden.

Protocol
: Jede Netzwerkschicht enthält meistens ein Protokoll-Feld damit die Übergabe transportierter Informationen an höhere Netzwerkschichten erleichtert wird. Beispiele hierfür: Schicht 2 (MAC) und 3 (IP).

Router
: Ein Router ist ein Knoten mit zwei (2) oder mehr (physischen oder virtuellen) Schnittstellen, der Datenpakete zwischen den Schnittstellen versenden kann.

Socket
: Ein IP socket wird durch Quell- und Zieladresse, den Ports (und der Verbindung) definiert.

Stack
: Ein Stack setzt sich aus Netzwerkschichten zusammen.

Subnetmask
: IP Netzwerke verwenden Bitmasken um lokale von entfernten Netzwerken zu trennen.

Tunnel
: Ein Tunnel ist typischerweise eine Punkt-zu-Punkt-Verbindung, über die Datenpakete eines anderen Protokolls ausgetauscht werden. Beispiel: IPv6-in-IPv4 Tunnel.

1.7.1.1. Abkürzungen

ACL
: Access Control List - Zugriffsliste)

API
: Application Programming Interface - Schnittstellen in Programmen zwischen den Applikationen

ASIC
: Application Specified Integrated Circuit - Applikationsspezifischer integrierter Schaltkreis

BSD
: Berkeley Software Distribution

CAN-Bus
: Controller Area Network Bus (physical bus system)

ISP
: Internet Service Provider

KAME

Ein Projekt und gemeinsame Anstrengung von sechs (6) Firmen in Japan mit dem Ziel, einen freien IPv6 und IPsec Stack für BSD Derivate der Öffentlichkeit zur Verfügung zu stellen www.kame.net.

LIR
: Local Internet Registry - Lokale Internet Registratur

NIC
: Network Interface Controller - Netzwerk[schnittstellen]karte, kurz Netzwerkkarte

RFC
: Request For Comments - eine Sammlung von technischen und organisatorischen Dokumenten zum Thema Internet.

USAGI
: UniverSAl playGround for Ipv6 Project - dieses Projekt will für das Linux System einen IPv6 Protokoll stack mit Produktionsqualität ausliefern.

1.7.2. In diesem Dokument verwendete Syntax

1.7.2.1. Zeilenumbruchs-Zeichen bei langen Codebeispielen

Das spezielle Zeichen "¬" zeigt in den Beispielen an, dass die Zeile umgebrochen wurde. Dies wurde für eine korrekte Darstellung des Textes in den PDF- und PS-Versionen benötigt.

1.7.2.2. Platzhalter

In allgemeinen Beispielen können Sie öfters lesen:

```
<myipaddress>
```

In Skripts oder an Ihrer Kommandozeile müssen Sie die < und > weglassen und den Text mit dem entsprechenden Inhalt ersetzen. Das Beispiel hier z.B. könnte sein:

```
1.2.3.4
```

1.7.2.3. Shell-Kommandos

Kommandos, die nicht als Root-Benutzer ausgeführt werden, beginnen mit $, z.B.

```
$ whoami
```

Befehle, die mit Root-Rechten ausgeführt werden, beginnen mit #, z.B.

```
# whoami
```

1.8. Grundvoraussetzung für die Verwendung dieses HOWTOs

1.8.1. Persönliche Anforderungen

1.8.1.1. Erfahrung mit Unix Tools

Sie sollten mit den gängigsten Unix Tools wie *grep*, *awk*, *find*, etc. und deren Kommandozeilen-Optionen vertraut sein.

1.8.1.2. Erfahrung mit Netzwerktheorie

Sie sollten das Schichtmodell und die einzelnen Schichten, Protokolle, Adressarten, Kabelsorten, Stecker etc. kennen. Wenn das Neuland für Sie sein sollte, finden Sie hier einen guten Ausgangspunkt: http://www.rigacci.org/docs/biblio/online/intro_to_networking/book1.htm

---

1.8.1.3. Erfahrung mit der Konfiguration von IPv4 Netzen

Sie sollten definitiv Erfahrung mit der Konfiguration von IPv4 Netzwerken haben, andernfalls werden Sie dem Text nur schwer folgen können.

1.8.1.4. Erfahrung mit dem Domain Name System (DNS)

Sie sollten ebenfalls das Domain Name System (DNS) verstehen und damit umgehen können.

1.8.1.5. Routine im Umgang mit Strategien zur Netzwerk-Fehlersuche

Sie sollten zumindest mit tcpdump umgehen und den Output des Programms interpretieren können. Andernfalls wird die Netzwerk-Fehlersuche für Sie schwierig.

1.8.2. Linux kompatible Hardware

Sicherlich wollen Sie mit realer Hardware experimentieren und nicht darüber an dieser Stelle lesen und an der einen oder anderen Stelle einschlummern. ;-7)

Kapitel 2. Grundlagen

2.1. Was ist IPv6?

IPv6 ist ein neues Schicht 3 Vermittlungsprotokoll und es wird IPv4 (auch als IP bekannt) ablösen. IPv4 wurde vor langer Zeit entworfen (RFC 760 / Internet Protocol vom Januar 1980). Seitdem wurden viele Adressen vergeben und Erweiterungen angeregt. Die aktuelle RFC ist RFC 2460 / Internet Protocol Version 6 Specification. Hauptänderungen in IPv6 sind das neue Design des Headers sowie die Erweiterung der Adresslänge von 32 bits auf 128 bits. Die Schicht 3 ist für den Transport der Pakete von Endpunkt-zu-Endpunkt mittels adressbasierten Paket-Routings zuständig, und wie bei IPv4 müssen bei IPv6 die Adressen (Quell- und Zieladresse) inkludiert sein.

Für weitere Informationen zur IPv6 Geschichte siehe die älteren RFCs z.B. SWITCH IPv6 Pilot / References.

2.2. Geschichte von IPv6 & Linux

Die Jahre 1992, 1993 und 1994 der allgemeinen IPv6 Geschichte können Sie in folgendem Dokument nachlesen: IPv6 or IPng (IP next generation).

Zu erledigen: Bessere Chronologie, mehr Inhalt

2.2.1. Anfang

Der erste IPv6 Netzwerk Code wurde dem Linux Kernel 2.1.8 im November 1996 durch Pedro Roque hinzugefügt. Er basierte auf dem BSD API:

```
diff -u --recursive --new-file v2.1.7/linux/include/linux/in6.h
¬ linux/include/linux/in6.h
--- v2.1.7/linux/include/linux/in6.h Thu Jan 1 02:00:00 1970
+++ linux/include/linux/in6.h Sun Nov 3 11:04:42 1996
@@ -0,0 +1,99 @@
+/*
+ * Types and definitions for AF_INET6
+ * Linux INET6 implementation
+ * + * Authors:
+ * Pedro Roque <******>
+ *
+ * Source:
+ * IPv6 Program Interfaces for BSD Systems
+ * <draft-ietf-ipngwg-bsd-api-05.txt>
```

Diese Zeilen entstammen dem patch-2.1.8 (die E-Mail-Adresse wurde hier beim Copy & Paste absichtlich gelöscht).

2.2.2. Übergangszeit

Aufgrund fehlender Arbeitskraft konnte die IPv6-Kernel-Implementierung nicht mit den Drafts oder neu freigegebenen RFCs Schritt halten. Im Oktober 2000 wurde in Japan das USAGI Projekt gestartet. Das Ziel war, die fehlende bzw. bereits veraltete IPv6 Funktionalität in Linux zu implementieren. Dabei richtete man sich nach der aktuellen FreeBSD Implementierung von IPv6, die durch das KAME project umgesetzt wurde. Von Zeit zu Zeit wurden im Vergleich zu den aktuellen Standard Linux-Kernel-Quellen ein Auszug erstellt.

Bis zum Start der Entwicklungs-Kernel Serie 2.5.x, der USAGI Patch war so groß, das er von den Linux-Netzwerkcode-Maintainers nicht komplett für die Einbindung in die Produktions-Kernel Serie 2.4.x eingebunden werden konnte.

Während der Entwicklung in der Serie 2.5.x hat USAGI versucht, so viel wie möglich ihrer Erweiterungen darin zu integrieren.

2.2.3. Heute

Viele der von USAGI und anderen lang entwickelten IPv6-bezogenen Patches sind bereits in der Vanilla Kernel Serie 2.6.x integriert.

2.2.4. Zukunft

USAGI und andere arbeiten weiterhin an der Implementierung von neuen Features wie Mobility und anderen. Von Zeit zu Zeit werden neue Erweiterungs-Patches veröffentlicht, wie auch die Integration in die Vanilla Kernel Serie 2.6.x vorangetrieben.

2.3. Wie sehen IPv6 Adressen aus?

Wie gesagt, IPv6 Adressen sind 128 bit lang. Diese bit-Anzahl kann sehr hohe dezimale Zahlen mit bis zu 39 Ziffern ergeben:

```
2^128-1: 340282366920938463463374607431768211455
```

Solche Zahlen sind nicht wirklich Adressen, die auswendig gelernt werden können. Die IPv6 Adressdarstellung ist bitweise orientiert (wie bei IPv4, aber das wird nicht oft bedacht). Eine bessere Schreibweise ist deshalb die hexadezimale Darstellung. Dabei werden 4 bits (auch "nibble" genannt) durch die Zeichen 0-9 und a-f (10-15) dargestellt, wodurch die Länge auf 32 Zeichen reduziert wird.

```
2^128-1: 0xffffffffffffffffffffffffffffffff
```

Diese Darstellung ist ebenfalls nicht sehr angenehm (mögliche Verwechslung oder Verlust einzelner hexadezimaler Ziffern), so dass die IPv6 Designer das hexadezimales Format mit einem Doppelpunkt als Trennzeichen nach jedem 16 bit Block erweiterten. Ferner wird das führende "0x" (ein in Programmiersprachen verwendetes Identifizierungsmerkmal für hexadezimale Werte) entfernt:

```
2^128-1: ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff
```

Eine gültige Adresse (s.u. Adress-Typen) ist z.B.:

```
2001:0db8:0100:f101:0210:a4ff:fee3:9566
```

Der Vereinfachung halber können führende Nullen jedes 16 bit-Blocks weggelassen werden:

```
2001:0db8:0100:f101:0210:a4ff:fee3:9566  ->
¬ 2001:0db8:100:f101:210:a4ff:fee3:9566
```

Eine Sequenz von 16 bit-Blöcken, die nur Nullen enthaltet, kann durch ein "::" ersetzt werden. Diese Komprimierung kann aber nicht öfters als einmal durchgeführt werden

```
2001:0db8:100:f101:0:0:0:1  ->  2001:0db8:100:f101::1
```

Die höchstmögliche Reduktion sieht man bei der IPv6 Localhost Adresse:

```
0000:0000:0000:0000:0000:0000:0000:0001  ->  ::1
```

Es gibt auch eine so genannte *kompakte* Darstellung (base 85 codiert) RFC 1924 / A Compact Representation of IPv6 Addresses (publiziert am 1. April 1996). Diese Notation wurde allerdings nie in der Praxis gesehen und ist wahrscheinlich ein Aprilscherz. Ein Beispiel:

```
# ipv6calc --addr_to_base85 2001:0db8:0100:f101:0210:a4ff:fee3:9566
Itu&-ZQ82s>J%s99FJXT
```

> Info: *ipv6calc* ist ein IPv6 Adressen-Format-Umrechner und Konvertierungsprogramm und ist hier zu finden: ipv6calc homepage (Mirror)

2.4. FAQ (Grundlagen)

2.4.1. Warum wird der Nachfolger von IPv4 nun IPv6 und nicht IPv5 genannt?

In jedem IP-Header werden die ersten 4 Bits für die Protokollversion reserviert. So sind theoretisch die Protokollnummern 0 bis 15 möglich:

- 4: Wird schon für IPv4 verwendet
- 5: Ist für das Stream Protocol (STP, RFC 1819 / Internet Stream Protocol Version 2) reserviert (das aber nie den Weg in die Öffentlichkeit fand)

So war die nächste freie Zahl 6. IPv6 war geboren!

2.4.2. IPv6 Adressen: Warum ist die Anzahl der Bits so groß?

Bei der Entwicklung von IPv4 dachte man, dass 32 Bits für die Welt ausreichend wären. Blickt man zurück, so waren bis heute 32 bits ausreichend. Vielleicht ist dies auch noch für ein paar Jahre so. Jedoch werden 32 bits nicht ausreichen, um in der Zukunft jedes Netzwerkgerät mit einer globalen Adresse ausstatten zu können. Denken Sie an Mobiltelefone, Autos (mit elektronischen Geräten an einem CAN Bus), Toaster, Kühlschränken, Lichtschalter usw.

Die IPv6 Designer haben 128 Bit gewählt, 4-mal mehr als im heutigen IPv4.

Aber die benutzbare Größe ist kleiner als es erscheinen mag, da in dem gegenwärtig definierten Adress-Schema 64 Bits für die Schnittstellen-Identifizierung verwendet werden. Die zweiten 64 Bit werden für das Routing verwendet. Die derzeitigen Aggregation-Levels (= Größe der zugeteilten IP-Blöcke) vorausgesetzt (/48, /32,...), ist eine Verknappung der Adressen weiterhin denkbar. Aber mit Sicherheit nicht in naher Zukunft.

Weitere Informationen finden Sie unter RFC 1715 / The H Ratio for Address Assignment Efficiency und RFC 3194 / The Host-Density Ratio for Address Assignment Efficiency.

2.4.3. IPv6 Adressen: Warum ist die Bit-Anzahl bei einem neuen Design so klein?

Es gibt (wahrscheinlich) eine Gruppe (bekannt ist nur Jim Fleming...) von Personen am Internet, die über IPv8 und IPv16 nachdenken. Für diese Designs gibt es aber keine hohe Akzeptanz und auch keine Kernel-Implementierungen. 128 bits sind die beste Wahl bezogen auf Header-Overhead und dem Datentransport. Denken Sie an die geringste Maximum Transfer Unit (MTU) in IPv4 (575 octets) und in IPv6 (1280 octets), die Header-Länge in IPv4 (20 octets Minimum, kann bis zu 60 octets mit IPv4 Optionen ansteigen) und in IPv6 sind es 48 octets (fixer Wert). Dies ist 3.4% der MTU in IPv4 und 3.8% der MTU in

IPv6. Dies bedeutet, dass der Overhead beim Header fast identisch ist. Mehr bits für die Adressierung würden größere Header und deshalb mehr Overhead erfordern. Bedenken Sie auch die maximale MTU von 1500 octets (in speziellen Fällen bei Jumbo-Paketen bis zu 9k octets) bei normalen Verbindungen (z.B. Ethernet). Letztlich wäre es kein korrektes Design, wenn 10% oder 20% der transportierten Daten in einem Schicht 3-Paket für Adressen und nicht für die "Nutzlast" benötigt würden.

Kapitel 3. Adress-Typen

Wie bei IPv4 können IPv6-Adressen mittels Subnetzmasken (subnet masks) in einen Netz- und einen Host-Teil unterteilt werden.

Bei IPv4 hat sich gezeigt, dass es manchmal von Nutzen wäre, einem Interface mehr als eine IP-Adresse zuweisen zu können, je nach Bedarf und Zweck (aliases, multicast etc.). Um in Zukunft flexibler bleiben zu können, geht man bei IPv6 weiter und erlaubt pro Interface mehr als eine zugewiesene IP-Adresse. Derzeit sind durch die RFCs kein Limit gesetzt, wohl aber in der Implementierung des IPv6 Stacks (um DoS Attacken vorzubeugen).

Neben der großen Bit-Anzahl für Adressen definiert IPv6 basierend auf einigen vorangestellten Bits verschiedene Adress-Typen. Diese werden hoffentlich in der Zukunft niemals aufgehoben (zum Unterschied zu IPv4 heute und die Entwicklung der class A, B und C Netze).

Zur Unterstützung einer automatischen Konfiguration wird die Bitanzahl in einen Netzwerk-Teil (vordere 64 Bits) und einen Hostteil (hintere 64 Bits).

3.1. Adressen ohne speziellen Präfix

3.1.1. Localhost Adresse

Dies ist eine spezielle Adresse für das Loopback Interface, vergleichbar zur "127.0.0.1" bei IPv4. Bei IPv6 lautet die localhost Adresse:

```
0000:0000:0000:0000:0000:0000:0000:0001
```

bzw. komprimiert:

```
::1
```

Pakete mit dieser Quell- bzw. Ziel-Adresse sollten niemals den sendenden Host verlassen.

3.1.2. Unspezifische Adresse

Dies ist eine spezielle Adresse vergleichbar mit "any" oder "0.0.0.0" bei IPv4. In IPv6 lautet sie:

```
0000:0000:0000:0000:0000:0000:0000:0000
```

oder:

```
::
```

Diese Adresse wird meistens in Routing-Tabellen und beim "socket binding" (zu jeder IPv6 Adresse) angewendet bzw. gesehen.

Beachten: Die Unspezifizierte Adresse kann nicht als Ziel-Adresse verwendet werden.

3.1.3. IPv6 Adressen mit eingebetteter IPv4 Adresse

Es gibt zwei Adressen-Typen, die IPv4 Adressen enthalten können.

3.1.3.1. IPv4 Adressen in IPv6 Format

IPv4-only IPv6-kompatible Adressen kommen manchmal bei IPv6 kompatiblen Daemon zur Anwendung, die allerdings ausschließlich an IPv4 Adressen gebunden sind.

Diese Adressen sind mit einer speziellen Präfixlänge von 96 definiert (a.b.c.d. ist die IPv4 Adresse):

```
0:0:0:0:0:ffff:a.b.c.d/96
```

oder in komprimiertem Format:

```
::ffff:a.b.c.d/96
```

Die IPv4 Adresse 1.2.3.4. z.B. sieht wie folgt aus:

```
::ffff:1.2.3.4
```

3.1.3.2. IPv4 kompatible IPv6 Adressen

Dieser Adress-Typ wurde für das automatische Tunneln (RFC 2893 / Transition Mechanisms for IPv6 Hosts and Routers) verwendet, welches aber durch das 6to4 tunneling ersetzt wurde.

```
0:0:0:0:0:0:a.b.c.d/96
```

oder in komprimierter Form:

```
::a.b.c.d/96
```

3.2. Netzteil der Adresse (Präfix)

Es wurden einige Adress-Typen definiert und zugleich blieb für zukünftige Anforderungen ausreichend Raum für weitere Definitionen. In RFC 4291 / IP Version 6 Addressing Architecture wird das aktuelle Adress-Schema definiert.

Lassen Sie uns nun einen Blick auf die verschiedenen Präfixe (und somit auf die Adress-Arten) werfen::

3.2.1. Link-lokaler Adress-Typ

Es handelt sich um spezielle Adressen, die ausschließlich auf einem Link eines Interfaces gültig sind. Wird diese Adresse als Zieladresse verwendet, so kann das Paket niemals einen Router passieren. Die Adresse wird bei der Link-Kommunikation eingesetzt, z.B.:

- Ist noch jemand anderer auf diesem Link?
- Ist jemand mit einer speziellen Adresse hier (z.B. Suche nach einem Router)?

Die Adresse beginnt mit (wobei *"x"* für ein hexadezimales Zeichen steht, im Normalfall *"0*")

```
fe8x:  <- zurzeit als einziger in Benutzung
fe9x:
feax:
febx:
```

Eine Adresse mit diesem Präfix gibt es an jedem IPv6 fähigen Interface nach einer stateless automatischen Konfiguration (dies ist der Regelfall).

3.2.2. Site-lokaler Adress-Typ

Diese Adressen sind vergleichbar zu den RFC 1918 / Address Allocation for Private Internets im heutigen IPv4. Eine Neuerung und Vorteil hierbei ist, vergleichbar zum 10.0.0.0/8 im IPv4, die Nutzbarkeit von 16 bits bzw. ein Maximum von 65536 Subnetzen.

Ein weiterer Vorteil: Da man bei IPv6 mehr als eine Adresse an ein Interface binden kann, ist auch die Zuweisung einer site-local Adresse zusätzlich zu einer globalen Adresse möglich.

Die Adresse beginnt mit:

```
fecx:  <- meistens genutzt.
fedx:
feex:
fefx:
```

(*"x"* ist ein hexadezimales Zeichen, normalerweise *"0*")

Dieser Adresstyp ist nun abgekündigt RFC 3879 / Deprecating Site Local Addresses und sollte nicht mehr verwendet werden. Für Tests im Labor sind solche Adressen meineserachtens aber immer noch eine gute Wahl.

3.2.3. Unique Local IPv6 Unicast Adressen

Weil die schon früh definierten site-local Adressen nicht eindeutig sind, kann dies zu großen Problemen führen, wenn z.B. einst unabhängige Netzwerke später zusammengeschlossen werden (Überlappung von Subnetzen). Aufgrund dessen und anderer Gründe wurde ein neuer Adresstyp definiert, genant RFC 4193 / Unique Local IPv6 Unicast Addresses.

Die Adresse beginnt mit:

```
fdxx:
fcxx:
```

Ein Teil des Präfix (40 Bits) werden pseudozufällig generiert. Es ist sehr unwahrscheinlich, daß zwei generierte Präfixe identisch sind.

Ein Beispiel für einen Präfix (generiert mit Hilfe des web-basierten Werkzeugs: Goebel Consult / createLULA):

```
fd0f:8b72:ac90::/48
```

3.2.4. Globaler Adress-Typ ("Aggregatable global unicast")

Heute gibt es ist per Definition eine globale Adress-Art (Das erste Design, "Provider based" genannt, wurde bereits vor einigen Jahren wieder aufgegeben RFC 1884 / IP Version 6 Addressing Architecture [obsolete]. Einige Überbleibsel hiervon sind in älteren Linux Kernelquellen noch zu finden.

Die Adresse beginnt mit (x sind hexadezimale Zeichen)

```
2xxx:
3xxx:
```

Hinweis: Der Zusatz "aggregatable" im Namen wird in aktuellen Drafts abgelegt. Es sind weitere Subarten definiert:

3.2.4.1. 6bone Test-Adressen

Diese globalen Adressen waren die Ersten definierten und auch benutzen Adressen. Sie alle beginnen mit:

```
3ffe:
```

Beispiel:

```
3ffe:ffff:100:f102::1
```

Eine spezielle 6bone Test-Adresse, die niemals weltweit einmalig ist, beginnt mit

```
3ffe:ffff:
```

und wird zumeist in alten Beispielen benutzt, um zu vermeiden, dass Anwender diese mit Copy & Paste in Ihre Konfigurationen übernehmen können. Auf diese Weise können Duplikate weltweit einmaliger Adressen aus Versehen bzw. Unachtsamkeit vermieden werden. Es würde für den Original-Host ernste Probleme bedeuten (z.B. Antwortpakete für niemals gesendete Anfragen bekommen...). Aufgrund dessen, daß IPv6 nun produktiv ist, wird dieser Präfix nicht mehr länger delegiert und nach dem 6.6.2006 vom Routing ausgenommen (mehr unter RFC 3701 / 6bone Phaseout).

3.2.4.2. 6to4 Adressen

Diese Adressen werden für einen speziellen Tunnelmechanismus verwendet [RFC 3056 / Connection of IPv6 Domains via IPv4 Clouds und RFC 2893 / Transition Mechanisms for IPv6 Hosts and Routers]. Sie kodieren eine gegebene IPv4 Adresse, ein eventuelles Subnetz und beginnen mit

```
2002:
```

z.B. wird 192.168.1.1/5 repräsentiert durch:

```
2002:c0a8:0101:5::1
```

Ein kleines Shell-Kommando kann aus einer IPv4 eine 6to4 Adresse erstellen:

```
ipv4="1.2.3.4"; sla="5"; printf "2002:%02x%02x:%02x%02x:%04x::1" `echo $ipv4
¬ | tr "." " "` $sla
```

Siehe auch tunneling using 6to4 und information about 6to4 relay routers.

3.2.4.3. Durch einen Provider zugewiesene Adressen für ein hierarchisches Routing

Diese Adressen werden an Internet Service Provider (ISP) delegiert und beginnen mit:

```
2001:
```

Präfixe für große ISPs (mit eigenem Backbone) werden durch local registries vergeben. Zurzeit wird ein Präfix mit der Länge 32 zugeteilt.

Grosse ISPs delegieren ihrerseits an kleinere ISPs ein Präfix mit der Länge 48.

3.2.4.4. Für Beispiele und Dokumentationen reservierte Adressen

Momentan sind zwei Adressbereiche für Beispiele und Dokumentationen RFC 3849 / IPv6 Address Prefix Reserved for Documentation reserviert:

```
3ffe:ffff::/32
2001:0DB8::/32   EXAMPLENET-WF
```

Diese Adressbereiche sollten nicht geroutet werden und am Übergangsrouter zum Internet (basierend auf Absendeadressen) gefiltert werden.

3.2.5. Multicast-Addressen

Multicast-Adressen werden für entsprechende Dienste verwendet.

Sie beginnen immer mit (xx ist hierbei der Wert der Reichweite)

```
ffxy:
```

Die Adressen werden in Reichweiten und Typen unterteilt:

3.2.5.1. Multicast-Bereiche

Die Multicast Reichweite ist ein Parameter, mit dem die maximale Distanz angegeben werden kann, die ein Multicast Paket sich von der versendenden Einheit entfernen kann.

Zurzeit sind folgende Regionen (reichweiten) definiert:

- ffx1: Node-lokal, Pakete verlassen niemals den Knoten
- ffx2: Link-lokal, Pakete werden niemals von Routers weitergeleitet, der angegebene Link wird nie verlassen.
- ffx5: Site-lokal, Pakete verlassen niemals den Standort (Site)
- ffx8: organisationsweit, Pakete verlassen niemals eine Organisation (nicht einfach zu implementieren, dies muss durch das Routing Protokoll abgedeckt werden)
- ffxe: Globale Reichweite
- Sonstige sind reserviert

3.2.5.2. Multicast-Typen

Es sind bereits viele Typen definiert bzw. reserviert (siehe RFC 4291 / IP Version 6 Addressing Architecture für weitere Details), einige Beispiele:

- All Nodes Adresse: ID = 1h, alle Hosts am lokalen Node (ff01:0:0:0:0:0:0:1) oder am angeschlossenen Link (ff02:0:0:0:0:0:0:1) werden adressiert.
- All Routers Adresse: ID = 2h, alle Router am lokalen Node (ff01:0:0:0:0:0:0:2), am angeschlossenen Link (ff02:0:0:0:0:0:0:2) oder am lokalen Standort werden adressiert.

3.2.5.3. Erforderliche node link-local Multicast Adresse

Diese spezielle Multicast Adresse wird als Zieladresse bei der Erkundung des Nahbereichs verwendet, da es ARP bei IPv6 im Gegensatz zu IPv4 nicht mehr gibt.

Ein Beispiel für diese Adresse könnte sein:

```
ff02::1:ff00:1234
```

Das benutzte Präfix zeigt, dass es sich um eine link-lokale Multicast Adresse handelt. Dass Suffix wird aus der Zieladresse erstellt. In diesem Beispiel soll ein Paket zur Adresse "fe80::1234" gesendet werden, aber die Netzwerk-Schicht hat keine Kenntnis der aktuellen Schicht 2 MAC Adresse. Die oberen 104 bits werde mit "ff02:0:0:0:01:ff00::/104" ersetzt und die unteres 24 bits bleiben unverändert. Diese Adresse wird nun "am Link" verwendet, um den entsprechenden Node zu finden, der wiederum seine Schicht 2 MAC Adresse als Antwort zurücksendet.

3.2.6. Anycast-Adressen

Anycast Adressen sind spezielle Adressen und werden verwendet, um besondere Bereiche wie den nächstgelegenen DNS-Server, den nächstliegenden DHCP Server und vergleichbare dynamische Gruppen abzudecken. Die Adressen werden dem Pool des Unicast Adressraums (global-aggregierbar oder Site-lokal zurzeit) entnommen. Der Anycast-Mechanismus (client view) wird von dynamischen Routing-Protokollen gehandhabt.

Hinweis: Anycast Adressen können nicht als Quelladresse verwendet werden, sondern ausschließlich als Zieladressen.

3.2.6.1. Subnet-Router Anycast-Adresse

Die Subnet-Router Anycast Adresse ist ein einfaches Beispiel für eine Anycast Adresse. Angenommen, der Knoten hat folgende global zugewiesene IPv6 Adresse:

```
2001:0db8:100:f101:210:a4ff:fee3:9566/64  <- Node's address
```

Die Subnet-Router Anycast Adresse wird durch komplette Streichung des Suffixes (die letzten gültigen 64 bits) erstellt:

```
2001:0db8:100:f101::/64  <- subnet-router anycast address
```

3.3. Adress-Typen (Host-Teil)

In Hinblick auf Auto-Konfigurations- und Mobilitätsfragen wurde entschieden, die niedrigeren 64 bits als Host-Bestandteil zu nutzen. Jedes einzelne Subnetz kann deshalb eine große Anzahl an Adressen enthalten.

Der Host-Teil kann aus unterschiedlichen Blickwinkeln betrachtet werden:

3.3.1. Automatisch erstellte Adressen (auch unter dem Namen stateless bekannt)

Bei der Auto-Konfiguration wird der Hostteil der Adresse durch die Konvertierung der MAC-Adresse eines Interfaces (falls vorhanden) zu einer einmaligen IPv6 Adresse (mittels EUI-64 Methode) generiert. Falls keine MAC-Adresse verfügbar ist (z.B. bei virtuellen Interfaces), wird anstelle dessen etwas anderes herangezogen (wie z.B. die IPv4 Adresse oder die MAC-Adresse eines physikalischen Interfaces).

Als Beispiel hat hier ein NIC folgende MAC-Adresse (48 bit):

```
00:10:A4:E3:95:66
```

Diese wird gemäß demIEEE-Tutorial EUI-64 Design für EUI-48 Identifiers zum 64 bit Interface Identifier erweitert:

```
0210:a4ff:fee3:9566
```

Mit einem gegebenen Präfix wird daraus die schon oben gezeigte IPv6-Adresse:

```
2001:0db8:0100:f101:0210:a4ff:fee3:9566
```

3.3.1.1. Datenschutzproblem mit automatisch erstellten Adressen sowie eine Lösung

Der "automatisch generierte" Hostteil ist weltweit einmalig (mit Ausnahme, wenn der Hersteller einer NIC die gleiche MAC-Adresse bei mehr als einer NIC einsetzt). Die Client-Verfolgung am Host wird dadurch möglich, solange kein Proxy verwendet wird.

Dies ist ein bekanntes Problem und eine Lösung wurde dafür definiert: Datenschutz-Erweiterung, definiert in RFC 3041 / Privacy Extensions for Stateless Address Autoconfiguration in IPv6 (es gibt bereits ein neueres Draft: draft-ietf-ipv6-privacy-addrs-v2-*). Es wird von Zeit zu Zeit mittels eines statischen und eines Zufallswertes ein neues Suffix erstellt. Hinweis: Dies ist nur für ausgehende Client-Verbindungen sinnvoll und bei bekannten Servern nicht wirklich sinnvoll.

3.3.2. Manuell festgelegte Adressen

Bei Servern ist es wahrscheinlich leichter, sich einfachere Adressen zu merken. Dies kann z.B. mit der Zuweisung einer zusätzlichen IPv6 Adresse an ein Interface geschehen.

```
2001:0db8:100:f101::1
```

Für das manuelle Suffix, wie "::1" im obigen Beispiel, muss das siebte höchstwertige Bit auf 0 gesetzt sein (das universale/local Bit des automatisch generierten Identifiers). Es sind auch noch andere (ansonsten nichtausgewählte) Bit-Kombinationen für Anycast-Adressen reserviert.

3.4. Präfixlängen für das Routing

Um eine maximale Reduktion an Routing-Tabellen zu erzielen, war in der frühen Design-Phase noch ein vollkommen hierarchischer Routing-Ansatz vorgesehen. Die Überlegungen hinter diesem Ansatz waren die gegenwärtigen IPv4 Routing-Einträge in den Haupt-Routern (mit über 104.000 Einträgen im Mai 2001) sowie die Reduktion des Speicherbedarfs für die Routing-Tabellen bei Hardware-Routern (ASIC "Application Specified Integrated Circuit", speziell konstuierter Chip) sowie ein daraus resultierender Geschwindigkeitszuwachs (weniger Einträge ergeben hoffentlich schnellere Abfragen).

Heutiger Standpunkt ist, dass das Routing für Netzwerke mit nur einem Service Provider hauptsächlich mit einem hierarchischen Design realisiert wird. Eine solche Vorgehensweise ist nicht möglich, wenn mehr als eine ISP-Verbindung besteht. Diese Problematik wird unter dem Thema multi-homing diskutiert (Infos zu multi-homing: drafts-ietf-multi6-*,IPv6 Multihoming Solutions).

3.4.1. Präfixlängen ("netmasks" genannt)

Vergleichbar zu IPv4, handelt es sich hierbei um den routbaren Netzwerkpfad für das stattfindende Routing. Da die Standard-Notierung der Netzmaske von 128 bit nicht sehr fein aussieht, verwenden die Designer das aus IPv4 bekannte Classless Inter Domain Routing Schema (CIDR, RFC 1519 / Classless Inter-Domain Routing). Mit Hilfe des CIDR wird die Bitanzahl der IP Adresse festgelegt, welche für das Routing verwendet werden. Diese Methode wird auch als "Slash"-Notation genannt.

Ein Beispiel:

```
2001:0db8:100:1:2:3:4:5/48
```

Diese Notation wird erweitert zu:

- Netzwerk:

```
2001:0db8:0100:0000:0000:0000:0000:0000
```

- Netzmaske:

```
ffff:ffff:ffff:0000:0000:0000:0000:0000
```

3.4.2. Zutreffende Routen

Im Normalfall (ohne QoS) ergibt eine Suche in der Routing-Tabelle eine Route mit der signifikantesten Adress-Bit-Anzahl, d.h. jene Route mit der größten Präfix-Länge wird zuerst herangezogen.

Wenn z.B. eine Routing-Tabelle folgende Einträge zeigt (Liste ist nicht komplett):

```
2001:0db8:100::/48     ::            U  1 0 0 sit1
2000::/3               ::192.88.99.1 UG 1 0 0 tun6to4
```

Die gezeigten Zieladressen der IPv6 Pakete werden über die entsprechenden Geräte geroutet

```
2001:0db8:100:1:2:3:4:5/48  ->  routed through device sit1
2001:0db8:200:1:2:3:4:5/48  ->  routed through device tun6to4
```

Kapitel 4. IPv6 System-Check

Bevor Sie IPv6 auf einem Linux Host einsetzen können, müssen sie überprüfen, ob das System IPv6 fähig ist. Eventuell haben Sie Änderungen vorzunehmen, um IPv6 zu ermöglichen.

4.1. IPv6 kompatibler Kernel

Neuere Linux Distributionen beinhalten bereits einen IPv6-fähigen Kernel. Die IPv6-Funktionalität wird im Allgemeinen als Modul kompiliert. Es ist aber durchaus möglich, dass das Modul nicht automatisch beim Start des Betriebssystems geladen wird.

Hinweis: Sie sollten die Kernel Serie 2.2.x nicht mehr verwenden, da die IPv6-Implementierung nicht mehr aktuell ist. Auch die in der Serie 2.4.x wird nicht mehr weiterentwickelt bzgl. der Definitionen in den neueren RFCs. Es wird empfohlen, einen aus der Serie 2.6.x zu verwenden.

4.1.1. Überprüfung der IPv6 Unterstützung im aktuellen Kernel

Um zu überprüfen, ob ihr aktueller Kernel IPv6 unterstützt, sollten sie einen Blick in ihr /proc-Dateisystem werfen. Folgende Einträge müssen existieren:

```
/proc/net/if_inet6
```

Einen kleinen automatischen Test können Sie wie folgt durchführen:

```
# test -f /proc/net/if_inet6 && echo "Running kernel is IPv6 ready"
```

Wenn dieser Test negativ verläuft, ist das IPv6 Modul aller Wahrscheinlichkeit noch nicht geladen.

4.1.2. IPv6 Module laden

Mit folgenden Befehl können Sie versuchen, das Modul zu laden:

```
# modprobe ipv6
```

Wenn dieser Befehl positiv verläuft, dann sollten Sie das Modul mit folgendem Befehl auflisten können:

```
# lsmod |grep -w 'ipv6' && echo "IPv6 module successfully loaded"
```

Der obige Test sollte nun erfolgreich verlaufen.

Hinweis: Ein Entfernen des Moduls im laufenden System wird derzeit nicht unterstützt und kann unter gewissen Bedingungen zu einem Absturz des Kernels führen.

4.1.2.1. Automatisches Laden des Moduls

Es ist möglich das IPv6 Modul bei Bedarf automatisch zu laden. Sie müssen nur folgende Zeile in die Konfigurationsdatei des kernel modul loaders eintragen (normalerweise: /etc/modules.conf oder /etc/conf.modules):

```
alias net-pf-10 ipv6  # automatically load IPv6 module on demand
```

Mit der folgenden Zeile ist es auch möglich, das automatische Laden des IPv6 Moduls auszuschalten.

```
alias net-pf-10 off   # disable automatically load of IPv6 module on demand
```

Anmerkung: In Kernel Series 2.6.x wurde der Modul-Lade-Mechanismus geändert. Die neue Konfigurationsdatei wird anstelle /etc/modules.conf nun /etc/modprobe.conf genannt.

4.1.3. Kernel-Kompilierung mit IPv6 Funktionalität

Wenn beide oben gezeigten Methoden ohne Erfolg blieben und ihr Kernel somit keine IPv6 Unterstützung bietet, dann haben Sie folgende Optionen:

- Aktualisieren Sie Ihre Distribution mit einer Version, die von Haus aus IPv6 unterstützt (empfohlen für Anfänger),
- Sie können einen Standard-Kernel kompilieren (einfach, wenn Sie die benötigten Optionen kennen)
- Kompilieren Sie die Kernel-Quellen ihrer Distribution (manchmal nicht ganz so einfach)
- Kompilieren Sie einen Kernel mit den USAGI-Erweiterungen

Falls Sie sich dazu entscheiden, einen neuen IPv6 kompatiblen Kernel zu kompilieren, sollten Sie auf jeden Fall bereits Erfahrung mit der Kernel-Kompilierung haben sowie das Linux Kernel HOWTO lesen.

Ein Vergleich zwischen dem Standard-Kernel und dem Kernel mit USAGI-Erweiterungen ist verfügbar unter IPv6+Linux-Status-Kernel.

4.1.3.1. Kompilieren eines Standard-Kernels

Detailliertere Ausführungen zur Kompilierung eines IPv6 fähigen Kernels finden Sie unter IPv6-HOWTO-2#kernel.

Hinweis: Sie sollten wann immer möglich die Kernel Serie 2.6.x oder höher einsetzen, da die IPv6 Unterstützung der Serie 2.4.x nur einige Backports erhält und die IPv6-Unterstützung von Serie 2.2.x hoffnungslos veraltet ist und nicht mehr weiterentwickelt wird.

4.1.3.2. Kompilieren eines Kernels mit USAGI-Erweiterungen

Wie für den Standard-Kernel gilt auch hier, dass das Kompilieren des Kernels nur fortgeschrittenen Benutzern empfohlen wird, die mit IPv6 und dem Kompilieren des Kernels bereits vertraut sind.

Siehe auch USAGI project / FAQ und Obtaining the best IPv6 support with Linux (Article) (Spiegel).

4.1.4. IPv6 kompatible Netzwerkgeräte

Nicht alle Netzwerkgeräte sind bereits (bzw. überhaupt) dazu in der Lage, IPv6 Pakete übertragen zu können. Den aktuellen Status können Sie unter IPv6+Linux-status-kernel.html#transport.

Ein entscheidender Punkt ist die Tatsache, dass ein IPv6 Paket wegen der Struktur der Netzwerkschicht in der Kernel-Implementierung nicht wirklich aufgrund der IP-Header-Nummer (6 anstelle 4) wiedererkannt wird. Es wird aufgrund der Protokollnummer der Schicht 2 Transport-Protokolls wiedererkannt. Folglich können IPv6 Pakete von keinem Transport-Protokoll verwendet werden, welche diese Protokoll-Nummer nicht nutzen. Hinweis: Das Paket wird nach wie vor über den Link transportiert, aber auf der Empfänger-Seite kann das Paket nicht verarbeitet werden (Sie können dies z.B. mit tcpdump sehen).

4.1.4.1. Gegenwärtig bekannte Verbindungsarten, die niemals IPv6 fähig sein werden

- Serial Line IP (RFC 1055 / SLIP), auch SLIPv4 genannt; das Gerät heißt: s1X
- Parallel Line IP (PLIP), gleich dem SLIP, Gerätename: plipX
- ISDN mit *rawip* Encapsulation; Gerätename: isdnX

4.1.4.2. Bekannte Verbindungsarten, die gegenwärtig IPv6 nicht unterstützen

- ISDN mit *syncppp* Encapsulation; Gerätename: ipppX (Designfrage des ipppd; in der Kernel Serie 2.5.x wird ipppX in einer allgemeinen PPP Schicht inkludiert)

4.2. IPv6 kompatible Tools zur Netzwerkkonfiguration

Ohne entsprechende Tools zur Konfiguration von IPv6 würden Sie mit einem IPv6 fähigen Kernel nicht weit kommen. Es gibt verschiedene Pakete womit IPv6 konfiguriert werden kann.

4.2.1. net-tools Paket

Das Paket net-tool beinhaltet einige Tools wie ifconfig und route. Mit den Tools kann IPv6 auf einem Interface konfiguriert werden. Sehen Sie sich die Ausgabe des Befehls *ifconfig -?* Bzw. *route -?* an. Finden Sie in der Ausgabe die Worte IPv6, inet6 oder Ähnliches, dann ist das Tool IPv6-kompatibel.

Automatische Überprüfung:

```
# /sbin/ifconfig -? 2>& 1|grep -qw 'inet6' && echo "utility 'ifconfig' is
¬ IPv6-ready"
```

Folgenden Check gibt es für route:

```
# /sbin/route -? 2>& 1|grep -qw 'inet6' && echo "utility 'route' is IPv6-ready"
```

4.2.2. iproute Paket

Alexey N.Kuznetsov (gegenwärtig ein Betreuer des Linux Network Codes) erstellte eine Tool-Sammlung, womit das Netzwerk mittels dem netlink Device konfiguriert wird. Diese Tool-Sammlung stellt mehr Funktionalität als das net-tools Paket zur Verfügung, ist aber nicht sehr umfangreich dokumentiert und nichts für schwache Nerven.

```
# /sbin/ip 2>&1 |grep -qw 'inet6' && echo "utility 'ip' is IPv6-ready"
```

Wird das Programm /sbin/ip nicht gefunden, dann wird die Installation des iproute Paketes empfohlen.

- Sie können dies (falls beinhaltet) von der benutzten Linux-Distribution installieren
- Sie können nach einem passenden RPM Paket unter RPMfind/iproute suchen (manchmal ist auch das kompilieren eines SRPMS Paketes zu empfohlen)

4.3. IPv6 Test/Debug-Programme

Nachdem Sie ihr System auf IPv6 vorbereitet haben, wollen Sie nun IPv6 für die Netzwerkkommunikation einsetzen. Zuerst sollten Sie lernen, IPv6 Pakete mit einem Sniffer Programm zu untersuchen. Dies ist zu empfehlen, denn in Hinblick auf Fehlersuche und Troubleshooting kann das Durchführen einer schnellen Diagnose von Nutzen sein.

4.3.1. IPv6 ping

Das Programm ist normalerweise im Paket *iputils* beinhaltet. Durch senden von ICMPv6 echo-request Paketen und warten auf ICMPv6 echo-reply Paketen können einfache Transport-Tests durchgeführt werden.

Anwendung

```
# ping6 <hostwithipv6address>
# ping6 <ipv6address>
# ping6 [-I <device>] <link-local-ipv6address>
```

Beispiel

```
# ping6 -c 1 ::1
PING ::1(::1) from ::1 : 56 data bytes
64 bytes from ::1: icmp_seq=0 hops=64 time=292 usec

--- ::1 ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max/mdev = 0.292/0.292/0.292/0.000 ms
```

Hinweis: ping6 benötigt direkten Zugriff auf den Socket und hierfür Root-Rechte. Wenn Nicht-Root-Benutzer ping6 nicht benutzen können, kann dies zwei Ursachen haben:

1. ping6 ist nicht im Pfad des Benutzers eingetragen; ping6 ist allgemein in /usr/sbin zu finden -> Lösung: Den Pfad ergänzen (nicht empfohlen)
2. ping6 lässt sich im Allgemeines wegen fehlender Root-Rechte nicht korrekt ausführen -> Lösung: chmod u+s /usr/sbin/ping6

4.3.1.1. Das Interface für einen IPv6 ping bestimmen

Wenn link-lokale Adressen für ein IPv6 ping verwendet werden, dann hat der Kernel keine Kenntnis darüber, durch welches (physikalische oder virtuelle) Gerät das Paket gesendet werden muss - jedes Gerät hat eine link-lokale Adresse. Ein Versuch resultiert in folgender Fehlermeldung:

```
# ping6 fe80::212:34ff:fe12:3456
connect: Invalid argument
```

In diesem Fall müssen Sie das Interface zusätzlich spezifizieren:

```
# ping6 -I eth0 -c 1 fe80::2e0:18ff:fe90:9205
PING fe80::212:23ff:fe12:3456(fe80::212:23ff:fe12:3456) from
¬ fe80::212:34ff:fe12:3478 eth0: 56 data bytes
64 bytes from fe80::212:23ff:fe12:3456: icmp_seq=0 hops=64 time=445 usec

--- fe80::2e0:18ff:fe90:9205 ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss round-trip
¬ min/avg/max/mdev = 0.445/0.445/0.445/0.000 ms
```

4.3.1.2. Ping6 zu Multicast-Adressen

Ein interessanter Mechanismus zum Aufspüren eines IPv6 aktiven Hosts am Link ist mit ping6 an eine link-lokale all-node Multicast Adresse zu pingen.

```
# ping6 -I eth0 ff02::1
PING ff02::1(ff02::1) from fe80:::2ab:cdff:feef:0123 eth0: 56 data bytes
```

```
64 bytes from ::1: icmp_seq=1 ttl=64 time=0.104 ms
64 bytes from fe80::212:34ff:fe12:3450: icmp_seq=1 ttl=64 time=0.549 ms (DUP!)
```

Bei IPv6 kann dieses Verhalten zurzeit, im Gegensatz zu IPv4, wo Antworten auf ein Ping auf die Broadcast Adresse unterdrückt werden können, nicht unterbunden werden. Ausnahme hierbei ist der Einsatz der lokalen IPv6 Firewall-Funktionalität.

4.3.2. IPv6 traceroute6

Dieses Programm ist normal im Paket iputils enthalten. Es ist ein Programm vergleichbar dem IPv4 traceroute. Unten sehen Sie ein Beispiel:

```
# traceroute6 www.6bone.net
traceroute to 6bone.net (3ffe:b00:c18:1::10) from 2001:0db8:0000:f101::2, 30
¬ hops max, 16 byte packets
 1 localipv6gateway (2001:0db8:0000:f101::1) 1.354 ms 1.566 ms 0.407 ms
 2 swi6T1-T0.ipv6.switch.ch (3ffe:2000:0:400::1) 90.431 ms 91.956 ms 92.377 ms
 3 3ffe:2000:0:1::132 (3ffe:2000:0:1::132) 118.945 ms 107.982 ms 114.557 ms
 4 3ffe:c00:8023:2b::2 (3ffe:c00:8023:2b::2) 968.468 ms 993.392 ms 973.441 ms
 5 3ffe:2e00:e:c::3 (3ffe:2e00:e:c::3) 507.784 ms 505.549 ms 508.928 ms
 6 www.6bone.net (3ffe:b00:c18:1::10) 1265.85 ms * 1304.74 ms
```

Anmerkung: Im Unterschied zu modernen IPv4 traceroute Versionen, welche den Einsatz von ICMPv4-echo-request Paketen wie auch UDP Paketen (default) ermöglichen, können mit IPv6-traceroute nur UDP Pakete versendet werden. Wie Sie vielleicht bereits wissen, werden von Firewalls bzw. von ACLs auf Routern ICMP echo-request Pakete mehr akzeptiert als UDP Pakete.

4.3.3. IPv6 tracepath6

Dieses Programm ist normalerweise im Paket *iputils* enthalten. Das Programm ist dem traceroute6 ähnlich, es gibt den Weg zu einem angegebenen Ziel wieder und misst hierbei den MTU-Wert. Unten sehen Sie ein Beispiel:

```
# tracepath6 www.6bone.net
 1?: [LOCALHOST] pmtu 1480
 1: 3ffe:401::2c0:33ff:fe02:14 150.705ms
 2: 3ffe:b00:c18::5 267.864ms
 3: 3ffe:b00:c18::5 asymm 2 266.145ms pmtu 1280
 3: 3ffe:3900:5::2 asymm 4 346.632ms
 4: 3ffe:28ff:ffff:4::3 asymm 5 365.965ms
 5: 3ffe:1cff:0:ee::2 asymm 4 534.704ms
 6: 3ffe:3800::1:1 asymm 4 578.126ms !N
Resume: pmtu 1280
```

4.3.4. IPv6 tcpdump

In Linux ist tcpdump ein Haupttool zum aufzeichnen von Paketen. Weiter unten sehen Sie einige Beispiele. Normalerweise ist die Ipv6-Unterstützung in der aktuellen Version 3.6 gegeben.

Bei tcpdump werden zur Geräuschminimierung bei der Paket-Filterung Ausdrücke eingesetzt:

- icmp6: ICMPv6 Datenverkehr wird gefiltert
- ip6: IPv6 Datenverkehr (inkl.ICMPv6) wird gefiltert
- proto ipv6: getunnelter IPv6-in-IPv4 Datenverkehr wird gefiltert
- not port ssh: zum unterdrücken der Anzeige von SSH Paketen während der Ausführung von tcpdump

bei einer remote SSH-Sitzung

Ebenfalls sind einige Kommandozeilen-Optionen sehr hilfreich, um detailliertere Informationen über die Pakete erlangen und protokollieren zu können. Für ICMPv6 Pakete sind hauptsächlich interessant:

- "-s 512": Bei der Aufzeichnung der Pakete wird die zu Aufzeichnungslänge auf 512 bytes vergrößert
- "-vv": wirklich sehr ausführliche Ausgabe
- "-n": Adressen werden nicht in Namen aufgelöst. Dies ist hilfreich, wenn die Reverse-DNS-Auflösung nicht sauber arbeiten sollte

4.3.4.1. IPv6 ping zur Adresse 2001:0db8:100:f101::1 über einen lokalen Link

```
# tcpdump -t -n -i eth0 -s 512 -vv ip6 or proto ipv6
tcpdump: listening on eth0
2001:0db8:100:f101:2e0:18ff:fe90:9205 > 2001:0db8:100:f101::1: icmp6: echo
¬ request (len 64, hlim 64)
2001:0db8:100:f101::1 > 2001:0db8:100:f101:2e0:18ff:fe90:9205: icmp6: echo
¬ reply (len 64, hlim 64)
```

4.3.4.2. IPv6 ping zur Adresse 2001:0db8:100::1 über einen IPv6-in-IPv4 Tunnel geroutet

1.2.3.4. und 5.6.7.8. sind Tunnel-Endpunkte (alle Adressen sind Beispiele)

```
# tcpdump -t -n -i ppp0 -s 512 -vv ip6 or proto ipv6
tcpdump: listening on ppp0
1.2.3.4 > 5.6.7.8: 2002:ffff:f5f8::1 > 2001:0db8:100::1: icmp6: echo request
¬ (len 64, hlim 64) (DF) (ttl 64, id 0, len 124)
5.6.7.8 > 1.2.3.4: 2001:0db8:100::1 > 2002:ffff:f5f8::1: icmp6: echo reply (len
¬ 64, hlim 61) (ttl 23, id 29887, len 124)
1.2.3.4 > 5.6.7.8: 2002:ffff:f5f8::1 > 2001:0db8:100::1: icmp6: echo request
¬ (len 64, hlim 64) (DF) (ttl 64, id 0, len 124)
5.6.7.8 > 1.2.3.4: 2001:0db8:100::1 > 2002:ffff:f5f8::1: icmp6: echo reply (len
¬ 64, hlim 61) (ttl 23, id 29919, len 124)
```

4.4. IPv6 kompatible Programme

Aktuelle Distributionen beinhalten bereits die gängigsten IPv6 kompatiblen Client- und Server-Programme. Weitere Infos gibt es unter IPv6+Linux-Status-Distribution. Falls ein Programm hier noch nicht gelistet sein sollte, können Sie unter IPv6 & Linux - Current Status - Applications nachlesen, ob das Programm bereits auf IPv6 portiert wurde und unter Linux bereits läuft. Für verbreitete Programme gibt es einige Hinweise unter IPv6 & Linux - HowTo - Part 3 und IPv6 & Linux - HowTo - Part 4.

4.5. IPv6 kompatible Client-Programme (Auswahl)

Um die folgend abgebildeten Tests durchzuführen, benötigen Sie ein funktionierendes IPv6 System. Bei einigen Beispielen werden Adressen angezeigt, die nur bei einer verfügbaren 6bone Verbindung erreichbar sind.

4.5.1. DNS-Überprüfung der IPv6 Adress-Auflösung

Jeder DNS-Server (Domain Name System) sollte aufgrund der Sicherheitsupdates der letzten Jahre bereits mit neuerer Software bestückt sein, die den Übergangs-IPv6-Adress-Standardtyp AAAA unterstützt (der neueste Standardtyp - A6 genannt - wird nur von BIND9 und höheren Versionen unterstützt und ist daher noch nicht allzu verbreitet. Ebenfalls nicht unterstützt wird die root Domain IP6.ARPA). Ein einfacher Test zum überprüfen der IPv6 Adress-Auflösung ist:

```
# host -t AAAA www.join.uni-muenster.de
```

Die Ausgabe des Tests sollte etwa wie folgt sein:

```
www.join.uni-muenster.de. is an alias for tolot.join.uni-muenster.de.
tolot.join.uni-muenster.de. has AAAA address
¬ 2001:638:500:101:2e0:81ff:fe24:37c6
```

4.5.2. IPv6 kompatible Telnet Client-Programme

IPv6 kompatible Clients sind verfügbar. Ein einfacher Test sieht wie folgt aus:

```
$ telnet 3ffe:400:100::1 80
Trying 3ffe:400:100::1...
Connected to 3ffe:400:100::1.
Escape character is '^]'.
HEAD / HTTP/1.0

HTTP/1.1 200 OK
Date: Sun, 16 Dec 2001 16:07:21
GMT Server: Apache/2.0.28 (Unix)
Last-Modified: Wed, 01 Aug 2001 21:34:42 GMT
ETag: "3f02-a4d-b1b3e080"
Accept-Ranges: bytes
Content-Length: 2637
Connection: close
Content-Type: text/html; charset=ISO-8859-1

Connection closed by foreign host.
```

Wird ein Text wie "cannot resolve hostname" ausgegeben, dann unterstützt der Telnet Client keine IPv6 Adressen.

4.5.3. IPv6 kompatible ssh Client-Programme

4.5.3.1. openssh

Aktuelle openssh-Versionen sind IPv6 kompatibel. Abhängig von der Konfiguration vor der Kompilierung gibt es zwei unterschiedliche Verhaltensweisen:

- --without-ipv4-default: Der Client versucht zuerst eine IPv6-Verbindung. Misslingt dies, wird eine IPv4-Verbindung aufgebaut
- --with-ipv4-default: standardmäßig wird eine IPv4-Verbindung aufgebaut. Eine IPv6-Verbindung muss, wie unten im Beispiel zu sehen ist, erzwungen werden:

```
$ ssh -6 ::1
user@::1's password: ******
[user@ipv6host user]$
```

Falls ihr ssh Client-Programm die Option "-6" nicht kennt, dann ist das Programm nicht IPv6 fähig. Dies ist bei den meisten ssh Paketen der Version 1 der Fall.

4.5.3.2. ssh.com

SSH.com's SSH Client und Server sind ebenfalls IPv6 kompatibel und darüber hinaus handelt es sich um freie Programme für die Linux- und FreeBSD-Plattform, unabhängig davon, ob sie zu kommerziellem oder zu persönlichen Zweck verwendet werden.

4.5.4. IPv6 kompatible Web-Browser

Einen aktuellen Statusüberblick zum Thema IPv6 kompatible Web-Browser ist unter IPv6+Linux-status-apps.html#HTTP verfügbar.

Die meisten Browser haben zurzeit noch ungelöste Probleme

1. Ist ein IPv4 Proxy in den Einstellungen eingetragen, dann werden IPv6 Anfragen zum Proxy gesendet. Der Proxy kann keine IPv6 Anfragen verstehen und somit scheitert die Anfrage. Lösung: Proxy Software aktualisieren (siehe weiter unten).
2. Automatik-Einstellungen des Proxy (*.pac) können aufgrund ihrer Beschaffenheit nicht derart erweitert werden, dass sie IPv6 Anfragen anders handhaben (z.B. kein Proxy verwenden) können (Sie sind in Javaskript geschrieben und ziemlich hard coded in den Quellen verankert; z.B. Maxilla Quellcode).

Ältere Browser-Versionen verstehen ebenfalls keine URL mit IPv6 Adressen wie z.B. http://[2001:a60:9002:1::186:6]/ (die angegebene URL funktioniert nur mit einem IPv6 kompatiblen Browser!).

Ein kleiner Test ist diese URL mit einem gegebenen Browser und ohne Proxy zu verwenden.

4.5.4.1. URLs zum testen

Ein guter Ausgangspunkt zum Betrachten von Webseiten mit IPv6 ist http://www.kame.net/. Ist die Schildkröte animiert, dann ist Verbindung mittels IPv6 Verbindung zustande gekommen, andererseits bleibt die Schildkröte statisch.

4.6. IPv6 kompatible Server

In diesem Teil des HOWTOs wird stärker auf Client-spezifische Belange eingegangen. Folglich sei zu IPv6 kompatiblen Servern wie sshd, httpd, telnetd usw. auf diese Stelle verwiesen: Hints for IPv6-enabled daemons.

4.7. FAQ (IPv6 Systemcheck)

4.7.1. Anwendung diverser Tools

4.7.1.1. Q: ping6 zu einer link-lokalen Adresse funktioniert nicht

Fehlermeldung: "*connect: Invalid argument*"

Der Kernel hat keine Kenntnis darüber, welchen physikalischen oder virtuellen Link Sie zum versenden von ICMPv6 Paketen verwenden möchten. Aus diesem Grund wird die Fehlermeldung ausgegeben.

Lösung: Spezifizieren Sie den Link, z.B.: "ping6 -I eth0 fe80::2e0:18ff:fe90:9205". Siehe auch program ping6 usage.

4.7.1.2. Q: ping6 oder traceroute6 funktioniert nicht als normaler Benutzer

Fehlermeldung: "*icmp socket: Operation not permitted*"

Diese Tools erzeugen spezielle ICMPv6 Pakete und versenden diese unter Verwendung von raw sockets im Kernel. Raw sockets können aber nur vom Benutzer "root" verwendet werden. Normale Benutzer bekommen aus diesem Grund diese Fehlermeldung.

Lösung: Wenn wirklich alle Benutzer auf diese Tools zugreifen sollen, können Sie dies mit setzen des "suid" bits mittels "chmod u+s / path/to/program" erreichen (siehe auch program ping6 usage). Falls nicht alle Benutzer das Programm benötigen, können Sie die Gruppenzugehörigkeit des Programms ändern, z.B. Gruppe "wheel". Fügen Sie alle Benutzer zu dieser Gruppe hinzu und entfernen Sie das execution bit für andere Benutzer mittels "chmod o-rwx /path/to/program". Alternativ können Sie auch "sudo" dazu verwenden, um Ihren Sicherheitsbestimmungen Rechnung zu tragen.

Kapitel 5. Interface-Konfiguration

5.1. Unterschiedliche Netzwerk-Geräte

Ein Knoten besitzt mehrere Netzwerk-Devices, die in Klassen zusammengefasst werden können:

- Physikalische Devices wie eth0, tr0
- Virtuelle Devices wie ppp0, tun0, tap0, sit0, isdn0, ippp0

5.1.1. Physikalische Devices

Physikalische Interfaces wie Ethernet oder Token-Ring bedürfen keiner speziellen Handhabung.

5.1.2. Virtuelle Devices

Virtuelle Interfaces hingegen benötigen immer eine spezielle Konfiguration.

5.1.2.1. IPv6-in-IPv4 Tunnel Interfaces

Diese Interfaces werden sitx genannt. Der Name sit ist eine Abkürzung für Simple Internet Transition. Das Gerät hat die Fähigkeit IPv6 Pakete in IPv4 Pakete zu verkapseln und diese dann über einen Tunnel zum entfernten Endpunkt zu transportieren.

sit0 hat eine spezielle Bedeutung: dieses Interface kann nicht für fest zugeordnete Tunnel verwendet werden.

5.1.2.2. PPP Interfaces

PPP Interfaces beziehen ihre IPv6 Funktionalität von einem IPv6 kompatiblen PPP Daemon.

5.1.2.3. ISDN HDLC Interfaces

Für HDLC mit IP encapsulation ist die IPv6 Funktionalität bereits im Kernel integriert.

5.1.2.4. ISDN PPP Interfaces

ISDN PPP Interfaces (ippp) werden durch den Kernel nicht mit IPv6 Funktionalität unterstützt. Es gibt auch keine Pläne hierfür, da im Kernel 2.5.+ dieser Interface-Typ durch eine allgemeinere ppp Interface Schicht ersetzt werden soll.

5.1.2.5. SLIP + PLIP

Wie bereits erwähnt, unterstützen diese Interfaces keinen IPv6 Transport (senden ist ok, das abfertigen ankommender Pakete funktioniert jedoch nicht).

5.1.2.6. Ether-tap Device

Ether-tap Devices sind IPv6 kompatibel und als stateless konfiguriert. Für den Gebrauch muss das Modul "ethertap" geladen werden.

5.1.2.7. tun Device

Nicht von mir getestet...

5.1.2.8. ATM

01/2002: ATM wird vom Standard-Kernel nicht, jedoch aber durch die USAGI-Erweiterungen unterstützt.

5.1.2.9. Sonstige

Wurde ein Interface vergessen...?

5.2. Interfaces ein/aus-schalten

Es gibt zwei Methoden, ein Interface ein- oder auszuschalten.

5.2.1. Verwendung von "ip"

Gebrauch:

```
# ip link set dev <interface> up
# ip link set dev <interface> down
```

Beispiel:

```
# ip link set dev eth0 up
# ip link set dev eth0 down
```

5.2.2. Verwendung von "ifconfig"

Gebrauch:

```
# /sbin/ifconfig <interface> up
# /sbin/ifconfig <interface> down
```

Beispiel:

```
# /sbin/ifconfig eth0 up
# /sbin/ifconfig eth0 down
```

Kapitel 6. IPv6 Adressen konfigurieren

Es gibt verschiedene Methoden zum konfigurieren einer IPv6 Adresse eines Interfaces. Sie können "ifconfig" oder "ip" dazu einsetzen.

6.1. Bestehende IPv6 Adressen anzeigen

Zuerst sollten sie überprüfen, ob und welche IPv6 Adressen bereits konfiguriert sind (etwa durch automatischer stateless Konfiguration).

6.1.1. Verwendung von "ip"

Anwendung:

```
# /sbin/ip -6 addr show dev <interface>
```

Beispiel für einen statisch konfigurierten Host:

```
# /sbin/ip -6 addr show dev eth0
2: eth0: <BROADCAST,MULTICAST,UP&gt; mtu 1500 qdisc pfifo_ fast qlen 100
inet6 fe80::210:a4ff:fee3:9566/10 scope link
inet6 2001:0db8:0:f101::1/64 scope global
inet6 fec0:0:0:f101::1/64 scope site
```

Beispiel für einen automatisch konfigurierten Host.

Hier sehen Sie einige automatisch konfigurierte IP Adressen und deren Lebensdauer.

```
# /sbin/ip -6 addr show dev eth0
3: eth0: <BROADCAST,MULTICAST,PROMISC,UP&gt; mtu 1500 qdisc pfifo_fast qlen
¬ 100
inet6 2002:d950:f5f8:f101:2e0:18ff:fe90:9205/64 scope global dynamic
valid_lft 16sec preferred_lft 6sec
inet6 3ffe:400:100:f101:2e0:18ff:fe90:9205/64 scope global dynamic
valid_lft 2591997sec preferred_lft 604797sec inet6 fe80::2e0:18ff:fe90:9205/10
¬ scope link
```

6.1.2. Verwendung von "ifconfig"

Anwendung:

```
# /sbin/ifconfig <interface>
```

Hier sehen Sie verschiedene IP Adressen mit unterschiedlichen Gültigkeitsbereichen (die Ausgabe wurde mit grep gefiltert)

```
# /sbin/ifconfig eth0 |grep "inet6 addr:"
inet6 addr: fe80::210:a4ff:fee3:9566/10 Scope:Link
inet6 addr: 2001:0db8:0:f101::1/64 Scope:Global
inet6 addr: fec0:0:0:f101::1/64 Scope:Site
```

6.2. Hinzufügen einer IPv6 Adresse

Die Vorgehensweise beim hinzufügen einer IPv6 Adresse ist vergleichbar mit dem "IP ALIAS"-Mechanismus bei IPv4 adressierten Interfaces.

6.2.1. Verwendung von "ip"

Anwendung:

```
# /sbin/ip -6 addr add <ipv6address>/<prefixlength> dev <interface>
```

Beispiel:

```
# /sbin/ip -6 addr add 2001:0db8:0:f101::1/64 dev eth0
```

6.2.2. Verwendung von "ifconfig"

Anwendung:

```
# /sbin/ifconfig <interface> inet6 add <ipv6address>/<prefixlength>
```

Beispiel:

```
# /sbin/ifconfig eth0 inet6 add 2001:0db8:0:f101::1/64
```

6.3. IPv6 Adressen entfernen

Diese Funktion wird selten benötigt. Vorsicht ist beim entfernen nicht existenter IPv6 Adressen geboten, da ältere Kernel dieses Fehlverhalten manchmal mit einem Crash quittieren.

6.3.1. Verwendung von "ip"

Anwendung:

```
# /sbin/ip -6 addr del <ipv6address>/<prefixlength> dev <interface>
```

Beispiel:

```
# /sbin/ip -6 addr del 2001:0db8:0:f101::1/64 dev eth0
```

6.3.2. Verwendung von "ifconfig"

Anwendung:

```
# /sbin/ifconfig <interface> inet6 del <ipv6address>/<prefixlength>
```

Beispiel:

```
# /sbin/ifconfig eth0 inet6 del 2001:0db8:0:f101::1/64
```

Kapitel 7. Konfiguration normaler IPv6-Routen

Wenn Sie Ihren lokalen Link verlassen und Pakete in das weltweite IPv6-Internet versenden wollen, dann benötigen Sie Routing. Wenn sich bereits ein IPv6 fähiger Router an Ihrem Link befindet, dann reicht eventuell das Hinzufügen von IPv6 Routen.

7.1. Bestehende IPv6-Routen anzeigen

Zuerst sollten sie überprüfen, ob und welche IPv6 Adressen bereits konfiguriert sind (etwa durch automatischer Konfiguration).

7.1.1. Verwendung von "ip"

Anwendung:

```
# /sbin/ip -6 route show [dev <device>]
```

Beispiel:

```
# /sbin/ip -6 route show dev eth0
2001:0db8:0:f101::/64 proto kernel metric 256 mtu 1500 advmss 1440
fe80::/10             proto kernel metric 256 mtu 1500 advmss 1440
ff00::/8              proto kernel metric 256 mtu 1500 advmss 1440
default               proto kernel metric 256 mtu 1500 advmss 1440
```

7.1.2. Verwendung von "route"

Anwendung:

```
# /sbin/route -A inet6
```

Sie sehen hier mehrere IPv6 Routen mit unterschiedlichen Adressen eines einzelnen Interfaces (bei der Ausgabe wurde das Interface eth0 herausgefiltert).

```
# /sbin/route -A inet6 |grep -w "eth0"
2001:0db8:0:f101 ::/64 :: UA  256 0 0 eth0 <- Interface route for global
¬ address
fe80::/10        ::       UA  256 0 0 eth0 <- Interface route for link-local
¬ address
ff00::/8         ::       UA  256 0 0 eth0 <- Interface route for all multicast
¬ addresses
::/0             ::       UDA 256 0 0 eth0 <- Automatic default route
```

7.2. Eine IPv6-Route über ein Gateway hinzufügen

Eine Route wird meistens benötigt, um mit IPv6 die Außenwelt über einen IPv6 fähigen Router und über Ihren Link zu erreichen.

7.2.1. Verwendung von "ip"

Anwendung:

```
# /sbin/ip -6 route add <ipv6network>/<prefixlength> via <ipv6address>
¬ [dev <device>]
```

Beispiel:

```
# /sbin/ip -6 route add 2000::/3 via 2001:0db8:0:f101::1
```

7.2.2. Verwendung von "route"

Anwendung:

```
# /sbin/route -A inet6 add <ipv6network>/<prefixlength> gw
¬ <ipv6address> [dev <device>]
```

Die optionale Angabe eines Devices wird dann benötigt, wenn die IPv6 Adresse des Gateways eine lokale Link-Adresse ist.

Im folgenden Beispiel wird eine Route für alle aktuellen globalen Adressen (2000::/3) über das Gateway 2001:0db8:0:f101::1 hinzugefügt.

```
# /sbin/route -A inet6 add 2000::/3 gw 2001:0db8:0:f101::1
```

7.3. Eine IPv6-Route über ein Gateway entfernen

Das manuelle entfernen einer Route wird nicht oft benötigt, meistens wird dies automatisch durch Netzwerk-Konfigurationsscripts beim herunterfahren (des Betriebssystems oder eines Interfaces) bewirkt.

7.3.1. Verwendung von "ip"

Anwendung:

```
# /sbin/ip -6 route del <ipv6network>/<prefixlength> via <ipv6address>
¬ [dev <device>]
```

Beispiel:

```
# /sbin/ip -6 route del 2000::/3 via 2001:0db8:0:f101::1
```

7.3.2. Verwendung von "route"

Anwendung:

```
# /sbin/route -A inet6 del <ipv6network>/<prefixlength> gw <ipv6address> [dev
¬ <device>]
```

Beispiel zum entfernen der im obigen Beispiel hinzugefügten Route:

```
# /sbin/route -A inet6 del 2000::/3 gw 2001:0db8:0:f101::1
```

7.4. Eine IPv6-Route über ein Interface hinzufügen

Diese Funktion wird manchmal im Fall dedizierter Punkt-zu-Punkt Verbindungen verwendet, in der Regel aber eher selten benötigt.

7.4.1. Verwendung von "ip"

Anwendung:

```
# /sbin/ip -6 route add <ipv6network>/<prefixlength> dev <device>
¬ metric 1
```

Beispiel:

```
# /sbin/ip -6 route add 2000::/3 dev eth0 metric 1
```

Der Metrik-Wert "1" wird verwendet, um mit dem Metrik Wert von route kompatibel zu sein; der Standard-Metrik-Wert von "ip" ist "1024".

7.4.2. Verwendung von "route"

Anwendung:

```
# /sbin/route -A inet6 add <network>/<prefixlength> dev <device>
```

Beispiel:

```
# /sbin/route -A inet6 add 2000::/3 dev eth0
```

7.5. Eine IPv6-Route über ein Interface entfernen

Dies wird manuell nicht so oft benötigt, jedoch aber beim herunterfahren von Konfigurationsscripts benutzt.

7.5.1. Verwendung von "ip"

Anwendung:

```
# /sbin/ip -6 route del <ipv6network>/<prefixlength> dev <device>
```

Beispiel:

```
# /sbin/ip -6 route del 2000::/3 dev eth0
```

7.5.2. Verwendung von "route"

Anwendung:

```
# /sbin/route -A inet6 del <network>/<prefixlength> dev <device>
```

Beispiel:

```
# /sbin/route -A inet6 del 2000::/3 dev eth0
```

7.6. FAQ für IPv6-Routen

7.6.1. Unterstützung einer IPv6 Default-Route

Ein Schwerpunkt bei IPv6 ist das hierarchische Routing. Aus diesem Grund werden in Routern nur wenige Routing-Einträge benötigt.

Einige Punkte sind im aktuellen Kernel zu beachten:

7.6.1.1. Clients (kein Routing eines Paketes!)

Ein client kann eine Default Route (z.B. "::/0") einrichten, diese aber auch durch automatische Konfiguration, z.B. mit radvd, erlernen:

```
# ip -6 route show | grep ^default
default via fe80::212:34ff:fe12:3450 dev eth0 proto kernel metric 1024 expires
¬ 29sec mtu 1500 advmss 1440
```

7.6.1.2. Router & Paketweiterleitung

Ältere Linux Kernel (zumindest <= 2.4.17) unterstützen keine Default Routen. Man kann dies einrichten, aber die Abfrage dieser Route misslingt im Fall, dass ein Paket weitergeleitet werden soll (normaler Zwecke eines Routers).

Falls ein entsprechender Kernel noch verwendet wird, kann "default routing" eingerichtet werden, wenn hierbei das einzig globale Adress-Präfix "2000::/3" verwendet wird.

Anmerkung: Walten Sie mit Vorsicht bei der Anwendung von default routing auf exponierten Routern, wenn keine Adressfilterung eingesetzt wird. Andernfalls kann Multicast- oder lokaler Site-Datenverkehr den Router ungewollt verlassen.

Kapitel 8. Neighbor Discovery

Die Neighbor Discovery (Ermittlung der Netzwerkumgebung) ist der IPv6 Nachfolger für das ARP (Address Resolution Protocol) bei IPv4. Sie können Informationen über die aktuelle Netzwerkumgebung gewinnen, Einträge erstellen und entfernen.

Der Kernel merkt sich erfolgreich gelernte "Nachbarn" (wie ARP in IPv4). Sie können die gelernten Einträge mit "ip" einsehen.

8.1. Netzwerkumgebung mit "ip" anzeigen

Mit dem folgenden Befehl können Sie die gelernten oder konfigurierten IPv6 Nachbarn anzeigen:

```
# ip -6 neigh show [dev <device>]
```

Das folgende Beispiel zeigt einen Nachbar, einen erreichbaren Router:

```
# ip -6 neigh show
fe80::201:23ff:fe45:6789 dev eth0 lladdr 00:01:23:45:67:89 router nud reachable
```

8.2. Tabelle der Netzwerkumgebung mit "ip" editieren

8.2.1. Eintrag manuell hinzufügen

Mit folgendem Befehl können Sie einen Eintrag manuell hinzufügen:

```
# ip -6 neigh add <IPv6 address> lladdr <link-layer address> dev <device>
```

Beispiel:

```
# ip -6 neigh add fec0::1 lladdr 02:01:02:03:04:05 dev eth0
```

8.2.2. Eintrag manuell entfernen

Sie können einen Eintrag auch löschen:

```
# ip -6 neigh del <IPv6 address> lladdr <link-layer address> dev <device>
```

Beispiel:

```
# ip -6 neigh del fec0::1 lladdr 02:01:02:03:04:05 dev eth0
```

8.2.3. Erweiterte Einstellungen

Das Tool "ip" ist weniger ausführlich dokumentiert, dennoch ist es sehr mächtig. Sehen Sie online mit "help" für weitere Details:

```
# ip -6 neigh help
Usage: ip neigh { add | del | change | replace } { ADDR [ lladdr LLADDR ]
          [ nud { permanent | noarp | stale | reachable } ]
          | proxy ADDR } [ dev DEV ]
       ip neigh {show|flush} [ to PREFIX ] [ dev DEV ] [ nud STATE ]
```

Es sieht aus, als seien manche Optionen ausschließlich für IPv4 gedacht... Es wird um Ihre Mithilfe gebeten, wenn Sie Informationen zu Optionen und der erweiterten Anwendung beisteuern können.

Kapitel 9. Konfiguration eines IPv6-in-IPv4 Tunnels

Wenn zum Verlassen des lokalen Netzwerks keine native IPv6-Anbindung vorhanden ist, wird zum Erreichen des weltweiten IPv6 Internet ein IPv6-in-IPv4 Tunnel benötigt.

Es gibt unterschiedliche Tunnel-Mechanismen sowie einige Möglichkeiten zum Einrichten eines Tunnels.

9.1. Tunnelarten

Es steht Ihnen mehr als eine Möglichkeit zur Verfügung, IPv6 Pakete über ausschließliche IPv4 Links zu tunneln.

9.1.1. Statische Punkt-zu-Punkt Tunnel: 6bone

Ein Punkt-zu-Punkt Tunnel ist ein dedizierter Tunnel zu einem Endpunkt, der Kenntnis über das lokale IPv6 Netzwerk (für das Routing zurück...) und die IPv4 Adresse des Tunnel-Endpunktes verfügt. Definition des Punkt-zu-Punkt Tunnels siehe: RFC 2893 / Transition Mechanisms for IPv6 Hosts and Routers.
Anforderungen:

- Die IPv4 Adresse des lokalen Tunnel-Endpunktes muss statisch sein, global eindeutig und vom entfernten Tunnel-Endpunkt aus erreichbar sein.
- Sie müssen ein globales IPv6 Präfix zugewiesen bekommen haben (siehe 6bone registry)
- Ein entfernter Tunnel-Endpunkt muss dazu in der Lage sein, ihr IPv6 Präfix bis zu Ihrem lokalen Tunnel-Endpunkt zu routen (wobei meistens manuelle Konfiguration notwendig wird).

9.1.2. Automatische Tunnel

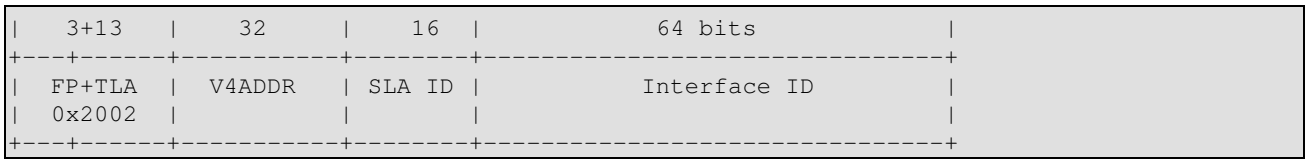
Automatisches Tunneln tritt dann ein, wenn ein Knoten direkt einen anderen Knoten (dessen IPv4-Adresse er zuerst kennen lernen muss) über die IPv4-mapped IPv6-Adresse anspricht.

9.1.3. 6to4 Tunnel

6to4 Tunnel (RFC 3056 / Connection of IPv6 Domains via IPv4 Clouds) verwenden einen einfachen Mechanismus zum erstellen eines automatischen Tunnels. Jeder Knoten mit einer weltweit einmaligen IPv4 Adresse kann zu einem 6to4 Tunnel-Endpunkt gemacht werden (solange keine IPv4-Firewall den Verkehr unterbindet). Ein 6to4 Tunnel ist zumeist kein one-to-one Tunnel. In diesem Fall wird das Untertunneln in einen Upstream- und einen Downstream-Tunnel unterteilt. Ferner zeigt eine spezielle IPv6 Adresse an, dass der Knoten einen 6to4 Tunnel für die Verbindung zum weltweiten IPv6 Netzwerk verwendet.

9.1.3.1. Erstellen eines 6to4 Präfixes

Die 6to4 Adresse wird wie folgt definiert (Schema ist dem RFC 3056 / Connection of IPv6 Domains via IPv4 Clouds entnommen):



FP und TLA zusammen haben den Wert 0x2002. V4ADDR ist die weltweit einmalige IPv4 Adresse des Knoten (in hexadezimaler Notation). Mit dem SLA wird das Subnetz identifiziert (65536 lokale Subnetze sind möglich) und benutzbar, um die lokale Netzwerstruktur abzubilden.

Für Gateways wird dieser Präfix normalerweise mit dem SLA "0000" definiert und dem 6to4 Tunnel-Interface das Suffix "::1" (kann aber auch ein beliebiger mit local-scope sein) zugewiesen. Zu bemerken ist, dass Microsoft Windows als Suffix auch immer die V4ADDR einsetzt.

9.1.3.2. 6to4 Tunnel zum Upstream

Der Knoten muss die Kenntnis darüber haben, an welchen entfernten Tunnel-Endpunkt die in IPv4 Paketen eingeschlossenen IPv6 Pakete gesendet werden sollen. In den "Anfängen" der 6to4 Tunnel-Anwendung wurden dedizierte Upstream akzeptierende Router definiert. Liste der Router siehe: NSayer's 6to4 information.

Heute können Upstream Router automatisch mittels der anycast Adresse 192.88.99.1 gefunden werden. Routing Protokolle sind für die Verarbeitung im Hintergrund zuständig, siehe RFC 3068 / An Anycast Prefix for 6to4 Relay Routers für weitere Details.

9.1.3.3. 6to4 Tunnel zum Downstream

Der Downstream (6bone -> Ihr 6to4 fähiger Node) ist nicht wirklich fix, er kann von jenem Host variieren, an dem ursprünglich die Pakete gesendet wurden. Es gibt zwei Möglichkeiten:

- Der entfernte Host benutzt 6to4 und sendet die Pakete direkt an den lokalen Knoten zurück (siehe unten).
- Der entfernte Host sendet die Pakete zurück an das weltweite IPv6 Netzwerk, und abhängig vom dynamischen Routing, erstellt dann ein Relay-Router automatisch zum lokalen Knoten einen Tunnel.

9.1.3.4. Möglicher 6to4 Verkehr

- Vom 6to4 zum 6to4: der Tunnel entsteht normalerweise direkt zwischen den beiden 6to4 fähigen Hosts.
- Vom 6to4 zum non-6to4: Der Datenstrom wird mittels Upstream-Tunnel versendet.
- Vom non-6to4 zum 6to4: Der Datenstrom wird mittels Downstream-Tunnel versendet.

9.2. Bestehende Tunnel anzeigen

9.2.1. Verwendung von "ip"

Anwendung:

```
# /sbin/ip -6 tunnel show [<device>]
```

Beispiel:

```
# /sbin/ip -6 tunnel show
sit0: ipv6/ip remote any local any ttl 64 nopmtudisc
sit1: ipv6/ip remote 195.226.187.50 local any ttl 64
```

9.2.2. Verwendung von "route"

Anwendung:

```
# /sbin/route -A inet6
```

Beispiel (Ausgabe wurde derart gefiltert, dass nur Tunnels über das virtuelle Interface sit0 angezeigt werden):

```
# /sbin/route -A inet6 | grep "\Wsit0\W*$"
::/96      ::               U   256  2  0  sit0
2002::/16  ::               UA  256  0  0  sit0
2000::/3   ::193.113.58.75  UG    1  0  0  sit0
fe80::/10  ::               UA  256  0  0  sit0
ff00::/8   ::               UA  256  0  0  sit0
```

9.3. Einrichtung eines Punkt-zu-Punkt Tunnels

Es gibt drei Methoden ein Punkt-zu-Punkt Tunnel hinzuzufügen bzw. zu entfernen.

Eine gute Informationsquelle zum Thema Tunnel-Einrichtung mit "ip" ist folgender Artikel: Configuring tunnels with iproute2 (article) (Spiegel).

9.3.1. Einen Punkt-zu-Punkt Tunnel hinzufügen

9.3.1.1. Verwendung von "ip"

Bei einer kleinen Anzahl von Tunnels ist die Verwendung von "ip" zurzeit die Standardmethode.

Beispiel für das Erstellen eines Tunnel-Devices (das Device wird aber hiermit nicht aktiviert; ebenso muss ein TTL Wert spezifiziert werden, da der Standardwert 0 ist):

```
# /sbin/ip tunnel add <device> mode sit ttl <ttldefault> remote
¬ <ipv4addressofforeigntunnel> local <ipv4addresslocal>
```

Anwendung (drei allgemeine Beispiele):

```
# /sbin/ip tunnel add sit1 mode sit ttl <ttldefault> remote
¬ <ipv4addressofforeigntunnel1> local <ipv4addresslocal>
# /sbin/ip link set dev sit1 up
# /sbin/ip -6 route add <prefixtoroute1> dev sit1 metric 1

# /sbin/ip tunnel add sit2 mode sit ttl <ttldefault>
¬ <ipv4addressofforeigntunnel2> local <ipv4addresslocal>
# /sbin/ip link set dev sit2 up
# /sbin/ip -6 route add <prefixtoroute2> dev sit2 metric 1

# /sbin/ip tunnel add sit3 mode sit ttl <ttldefault>
¬ <ipv4addressofforeigntunnel3> local <ipv4addresslocal>
# /sbin/ip link set dev sit3 up
# /sbin/ip -6 route add <prefixtoroute3> dev sit3 metric 1
```

9.3.1.2. Verwendung von "ifconfig" und "route" (nicht empfehlenswert)

Diese Methode zum Hinzufügen eines Tunnels wird nicht empfohlen, da Ungereimtheiten auftreten. Es gibt keine Probleme, wenn nur ein Tunnel hinzugefügt wird. Werden hingegen mehrere Tunnel hinzugefügt, dann kann der erste Tunnel nicht einfach deaktiviert werden, wenn die restlichen Tunnel aktiviert bleiben sollen.

Anwendung (drei allgemeine Beispiele):

```
# /sbin/ifconfig sit0 up

# /sbin/ifconfig sit0 tunnel <ipv4addressofforeigntunnel1>
# /sbin/ifconfig sit1 up
# /sbin/route -A inet6 add <prefixtoroute1> dev sit1

# /sbin/ifconfig sit0 tunnel <ipv4addressofforeigntunnel2>
# /sbin/ifconfig sit2 up
# /sbin/route -A inet6 add <prefixtoroute2> dev sit2

# /sbin/ifconfig sit0 tunnel <ipv4addressofforeigntunnel3>
# /sbin/ifconfig sit3 up
# /sbin/route -A inet6 add <prefixtoroute3> dev sit3
```

WICHTIG: NICHT VERWENDEN! Mit diesem Setup wird von überall aus dem Internet das "automatische Tunneln" vorbehaltlos aktiviert. Das ist ein unnötiges Risiko.

9.3.1.3. Verwendung allein von "route"

Sie können einen Tunnel auch im NBMA-Stil (Non Broadcast Multiple Access) einrichten. Bei dieser Vorgehensweise können Sie sehr einfach mehrere Tunnels zugleich einrichten, aber kein Tunnel kann nummeriert werden (und das ist ein kein benötigtes Feature).

Anwendung (drei allgemeine Beispiele):

```
# /sbin/ifconfig sit0 up

# /sbin/route -A inet6 add <prefixtoroute1> gw
¬ ::<ipv4addressofforeigntunnel1> dev sit0
# /sbin/route -A inet6 add <prefixtoroute2> gw
¬ ::<ipv4addressofforeigntunnel2> dev sit0
# /sbin/route -A inet6 add <prefixtoroute3> gw
¬ ::<ipv4addressofforeigntunnel3> dev sit0
```

WICHTIG: NICHT VERWENDEN! Mit diesem Setup wird von überall aus dem Internet das "automatische Tunneln" vorbehaltlos aktiviert. Das ist ein unnötiges Risiko.

9.3.2. Punkt-zu-Punkt Tunnel entfernen

Diese Funktion wird selten manuell durchgeführt. Skripte verwenden diese Funktion zum sauberen deaktivieren bzw. beim Neustart einer IPv6 Konfiguration.

9.3.2.1. Verwendung von "ip"

Entfernen eines Tunnel-Devices:

```
# /sbin/ip tunnel del <device>
```

Anwendung (drei allgemeine Beispiele):

```
# /sbin/ip -6 route del <prefixtoroute1> dev sit1
# /sbin/ip link set sit1 down
# /sbin/ip tunnel del sit1

# /sbin/ip -6 route del <prefixtoroute2> dev sit2
# /sbin/ip link set sit2 down
# /sbin/ip tunnel del sit2

# /sbin/ip -6 route del <prefixtoroute3> dev sit3
# /sbin/ip link set sit3 down
# /sbin/ip tunnel del sit3
```

9.3.2.2. Verwendung von "ifconfig" und "route" (nicht empfehlenswert, da unbequem)

Nicht nur bei der Erstellung eines Tunnels kommt es zu Ungereimtheiten, sondern auch bei dessen Entfernung. Die Tunnel müssen in umgekehrter Reihenfolge wieder entfernt werden, d.h. der zuletzt erstellte Tunnel muss als Erster entfernt werden...

Anwendung (drei allgemeine Beispiele):

```
# /sbin/route -A inet6 del <prefixtoroute3> dev sit3
# /sbin/ifconfig sit3 down

# /sbin/route -A inet6 del <prefixtoroute2> dev sit2
# /sbin/ifconfig sit2 down

# /sbin/route -A inet6 add <prefixtoroute1> dev sit1
# /sbin/ifconfig sit1 down

# /sbin/ifconfig sit0 down
```

9.3.2.3. Verwendung von "route"

Die Vorgehensweise ist vergleichbar mit dem löschen einer normalen IPv6 Route.

Anwendung (drei allgemeine Beispiele):

```
# /sbin/route -A inet6 del <prefixtoroute1> gw
¬ ::<ipv4addressofforeigntunnel1> dev sit0
# /sbin/route -A inet6 del <prefixtoroute2> gw
¬ ::<ipv4addressofforeigntunnel2> dev sit0
# /sbin/route -A inet6 del <prefixtoroute3> gw
¬ ::<ipv4addressofforeigntunnel3> dev sit0

# /sbin/ifconfig sit0 down
```

9.3.3. Nummerierte Punkt-zu-Punkt Tunnel

Manchmal ist es notwendig, einen Punkt-zu-Punkt Tunnel mit IPv6 Adresse genauso einzurichten, wie heute bei IPv4. Dies ist nur mit der ersten (ifconfig+route - nicht empfehlenswert) sowie mit der dritten (ip+route) beschriebenen Methode zur Einrichtung eines Tunnels möglich. Bei diesen Fällen können Sie den Tunnel-Interfaces die IPv6 Adressen, wie im Abschnitt zur Interface-Konfiguration beschrieben, hinzufügen.

9.4. Einrichtung von 6to4 Tunnel

Beachten sie Bitte, dass 6to4 Tunnel im Standard-Kernel der Serie 2.2.x (siehe systemcheck/kernel) nicht unterstützt werden. Weiter ist zu beachten, dass die Präfix-Länge für 6to4 Adressen 16 ist, da sich aus Perspektive des Netzwerks betrachtet, alle anderen 6to4 Hosts sich in der gleichen Schicht 2 befinden.

9.4.1. 6to4 Tunnel hinzufügen

Zu Anfang müssen Sie Ihre 6to4 Präfix-Länge mittels der lokal zugewiesenen global routbaren IPv4 Adresse berechnen (sollte ihr Host keine global routebare IPv4 Adresse haben, dann ist unter speziellen Bedingungen NAT auf dem Border Gateway möglich):

Angenommen, Ihre IPv4 Adresse ist:

```
1.2.3.4
```

Dann ist das daraus resultierende 6to4 Präfix:

```
2002:0102:0304::
```

Lokalen 6to4 Gateways sollte immer (ist aber kein Muss, ein beliebiger local-scope Suffix kann benutzt werden) das Suffix "::1" zugewiesen werden. Daraus resultierend ergibt sich Ihre lokale 6to4 Adresse:

```
2002:0102:0304::1
```

Zum automatischen Erstellen der Adresse können Sie folgenden Befehl nutzen:

```
ipv4="1.2.3.4"; printf "2002:%02x%02x:%02x%02x::1" `echo $ipv4 | tr "." " "`
```

Es gibt nun zwei Möglichkeiten einen 6to4 Tunnel einzurichten.

9.4.1.1. Verwendung von "ip" und einem dedizierten Tunnel-Device

Die empfohlene Vorgehensweise (der Wert TTL muss angegeben werden, da der Standardwert 0 ist):

Erstellen eines neues Tunnel-Device:

```
# /sbin/ip tunnel add tun6to4 mode sit ttl <ttldefault> remote any local
¬ <localipv4address>
```

Interface aktivieren:

```
# /sbin/ip link set dev tun6to4 up
```

Eine lokale 6to4 Adresse am Interface hinzufügen (Hinweis: Präfix-Länge 16 ist wichtig!)

```
# /sbin/ip -6 addr add <local6to4address>/16 dev tun6to4
```

Hinzufügen der (Standard-) Route zum globalen IPv6 Netz unter Verwendung der all-6to4-routers IPv4 anycast Adresse:

```
# /sbin/ip -6 route add 2000::/3 via ::192.88.99.1 dev tun6to4 metric 1
```

Manche Versionen von "ip" (z.B. SuSE Linux 9.0) unterstützen keine IPv4-kompatiblen IPv6-Adressen für Gateways, in diesem Fall muss die entsprechende IPv6-Adresse benutzt werden:

```
# /sbin/ip -6 route add 2000::/3 via 2002:c058:6301::1 dev tun6to4 metric 1
```

9.4.1.2. Verwendung von "ifconfig" und "route" sowie einem generischen Tunnel-Device "sit0" (nicht empfehlenswert)

Diese Vorgehensweise wird nicht empfohlen, da bei Verwendung des allgemeinen Tunnel Device sit0 keine Filter-Spezifizierung pro Device ermöglicht wird.

Das allgemeine Tunnel Interface sit0 aktivieren:

```
# /sbin/ifconfig sit0 up
```

Dem Interface eine lokale 6to4 Adresse hinzufügen:

```
# /sbin/ifconfig sit0 add <local6to4address>/16
```

Hinzufügen der (Standard-) Route zum globalen IPv6 Netz unter Verwendung der all-6to4-routers IPv4 anycast Adresse:

```
# /sbin/route -A inet6 add 2000::/3 gw ::192.88.99.1 dev sit0
```

9.4.2. 6to4 Tunnel entfernen

9.4.2.1. Verwendung von "ip" und einem dedizierten Tunnel-Device

Entfernen aller Routen über dieses bestimmten Tunnel Devices:

```
# /sbin/ip -6 route flush dev tun6to4
```

Interface deaktivieren:

```
# /sbin/ip link set dev tun6to4 down
```

Ein erstelltes Tunnel Device entfernen:

```
# /sbin/ip tunnel del tun6to4
```

9.4.2.2. Verwendung von "ifconfig" und "route" sowie einem generischen Tunnel-Device "sit0" (nicht empfehlenswert)

Entfernen der (Standard-) Route über ein 6to4 Tunnel Device:

```
# /sbin/route -A inet6 del 2000::/3 gw ::192.88.99.1 dev sit0
```

Eine 6to4 Adresse des Interfaces entfernen:

```
# /sbin/ifconfig sit0 del <local6to4address>/16
```

Ein allgemeines Tunnel Device deaktivieren (aber Achtung, eventuell ist das Interface noch in Verwendung...)

```
# /sbin/ifconfig sit0 down
```

Kapitel 10. IPv4-in-IPv6 Tunnel konfigurieren

RFC 2473 / Generic Packet Tunneling in IPv6 Specification spezifiziert den Mechanismus, um unterschiedliche Pakettypen (einschließlich IPv4) über IPv6 zu tunneln.

ANMERKUNG: Unterstützung für IPv4-in-IPv6 Tunnel ist erst seit Kernel Version 2.6.22 verfügbar.

10.1. Anzeigen von existierenden Tunnels

Anwendung:

```
# /sbin/ip -6 tunnel show [<device>]
```

Beispiel:

```
# /sbin/ip -6 tunnel show mode any
ip6tnl0: ipv6/ipv6 remote :: local :: encaplimit 0 hoplimit 0 tclass 0x00 flowlabel 0x00000 (flow
ip6tnl1: ip/ipv6 remote fd00:0:0:2::a local fd00:0:0:2::1 dev eth1 encaplimit 4 hoplimit 64 tclas
```

ANMERKUNG: wenn "mode any" nicht angegeben wird, werde nur IPv6-in-IPv6 Tunnels angezeigt.

10.2. Konfiguration eines Punkt-zu-Punkt Tunnels

Anwendung für die Erzeugung einer 4over6 Tunnel-Schnittstelle (welche danach aber noch nicht aktiv ist)

```
# /sbin/ip tunnel add <device> mode ip4ip6 remote <ipv6addressofforeigntunnel> local <ipv6address
```

Anwendung (allgemeines Beispiel für drei Tunnels):

```
# /sbin/ip -6 tunnel add ip6tnl1 mode ip4ip6 remote <ipv6addressofforeigntunnel1> local <ipv6addr
# /sbin/ip link set dev ip6tnl1 up
# /sbin/ip -6 route add <prefixtoroute1> dev ip6tnl1 metric 1

# /sbin/ip -6 tunnel add ip6tnl2 mode ip4ip6 remote <ipv6addressofforeigntunnel2> local <ipv6addr
# /sbin/ip link set dev ip6tnl2 up
# /sbin/ip -6 route add <prefixtoroute2> dev ip6tnl2 metric 1

# /sbin/ip -6 tunnel add ip6tnl3 mode ip4ip6 remote <ipv6addressofforeigntunnel3> local <ipv6addr
# /sbin/ip link set dev ip6tnl3 up
# /sbin/ip -6 route add <prefixtoroute3> dev ip6tnl3 metric 1
```

10.3. Löschen von Punkt-zu-Punkt-Tunnels

Anwendung für das Löschen einer Tunnel-Schnittstelle:

```
# /sbin/ip -6 tunnel del <device>
```

Anwendung (allgemeines Beispiel für drei Tunnels):

```
# /sbin/ip -6 route del <prefixtoroute1> dev ip6tnl1
# /sbin/ip link set ip6tnl1 down
# /sbin/ip -6 tunnel del ip6tnl1

# /sbin/ip -6 route del <prefixtoroute2> dev ip6tnl2
# /sbin/ip link set ip6tnl2 down
# /sbin/ip -6 tunnel del ip6tnl2
```

```
# /sbin/ip -6 route del <prefixtoroute3> dev ip6tnl3
# /sbin/ip link set ip6tnl3 down
# /sbin/ip -6 tunnel del ip6tnl3
```

Kapitel 11. Kernel-Einstellungen im /proc-Dateisystem

Anmerkung: Dieses Kapitel basiert großteils auf der Datei "ip-sysctl.txt", welche in den aktuellen Kernel-Quellen im Verzeichnis "Documentation/networking" zu finden ist. Danke an dieser Stelle an Pekka Savola, der den IPv6 relevanten Inhalt dieser Datei wartet und betreut. Ebenso sei erwähnt, dass einige Textstellen hieraus mehr oder weniger mit Copy & Paste in dieses Dokument übernommen wurden.

11.1. Zugriff auf das /proc-Dateisystem

11.1.1. Verwendung von "cat" und "echo"

Mit "cat" und "echo" können Sie am einfachsten das /proc Dateisystem einsehen. Hierfür gibt es aber einige Voraussetzungen, die erfüllt sein müssen:

- Das /proc-Dateisystem muss im Kernel aktiviert sein. Hierfür muss die folgende Einstellung beim kompilieren des Kernels vorgenommen werden:

```
CONFIG_PROC_FS=y
```

- Das /proc-Dateisystem muss zuerst gemountet sein. Dies kann wie folgt getestet werden:

```
# mount | grep "type proc"
none on /proc type proc (rw)
```

- Sie benötigen Lese- und manchmal auch Schreib-Zugriff (normalerweise nur als Root-Benutzer) auf das /proc-Dateisystem.

Normalerweise sind, mit Ausnahme in /proc/sys/*, alle Einträge ausschließlich mit Leserechten ausgestattet. Die Einträge werden zum Zweck der Informationsgewinnung verwendet.

11.1.1.1. Wert anzeigen

Den Inhalt eines Eintrags können sie mit "cat" anzeigen:

```
# cat /proc/sys/net/ipv6/conf/all/forwarding
0
```

11.1.1.2. Wert einstellen

Mit "echo" können sie einen neuen Wert zuweisen (nur wenn der Eintrag beschreibbar ist):

```
# echo "1" >/proc/sys/net/ipv6/conf/all/forwarding
```

11.1.2. Verwendung von "sysctl"

Die Verwendung des Programms "sysctl" ist eine zeitgemäße Methode zum Anzeigen der Kernel-Switches. Es funktioniert auch dann, wenn das /proc-Dateisystem nicht gemountet ist, wobei aber nur ein Zugriff auf /proc/sys/* möglich ist!

Das Programm "sysctl" ist (auf Red Hat Linux Systemen) im Paket "procps" enthalten.

- Das sysctl-Interface muss im Kernel aktiviert sein. Hierfür muss die folgende Einstellung beim kompilieren des Kernels vorgenommen werden:

```
CONFIG_SYSCTL=y
```

11.1.2.1. Wert anzeigen

Der Wert eines Eintrags kann nun angezeigt werden:

```
# sysctl net.ipv6.conf.all.forwarding
net.ipv6.conf.all.forwarding = 0
```

11.1.2.2. Wert einstellen

Ein neuer Wert kann wie folgt zugewiesen werden (wenn der Eintrag beschreibbar ist):

```
# sysctl -w net.ipv6.conf.all.forwarding=1
net.ipv6.conf.all.forwarding = 1
```

Anmerkung: Verwenden Sie beim setzen eines Wertes keine Leerzeichen vor oder nach dem "=". Sollten Sie mehrere Werte in einer Zeile angeben, müssen sie diese mit Anführungszeichen umgeben:

```
# sysctl -w net.ipv4.ip_local_port_range="32768 61000"
net.ipv4.ip_local_port_range = 32768 61000
```

11.1.2.3. Sonstiges

Anmerkung: Es gibt sysctl-Versionen im Umlauf, die anstelle des Punktes "." einen slash "/" ausgeben.

Für weitere Details siehe die manpage von sysctl.

Hinweise: Um schnell einen Überblick über die Einstellungen zu bekommen, verwenden Sie einfach die Option "-a" (anzeigen aller Einträge) sowie das Tool "grep".

11.1.3. Werte im /proc-Dateisystem

Es gibt im /proc-Dateisystem unterschiedliche Formate:

- BOOLEAN: einfach eine "0" (falsch) oder eine "1" (wahr)
- INTEGER: Wert ist eine Ganzzahl (kann auch eine unsigned int sein)
- Kompliziertere Zeilen mit verschiedenen Werten: manchmal wir eine Header-Zeile mit angezeigt... Sie können aber auch weitere Informationen zu den Werten und deren Bedeutung direkt in den Kernel Quellen beziehen.

11.2. Einträge in /proc/sys/net/ipv6/

11.2.1. conf/default/*

Ändern der Interface-spezifischen Einstellungen.

11.2.2. conf/all/*

Ändern aller Interface-spezifischen Einstellungen.

Ausnahme: "conf/all/forwarding" hat hier eine andere Bedeutung:

11.2.2.1. conf/all/forwarding

- Typ: BOOLEAN

Hiermit wird die globale IPv6 Weiterleitung zwischen allen Interfaces aktiviert.

In IPv6 ist kein forwarding per Device möglich. Die Steuerung der Weiterleitung muss mittels IPv6-netfilter Regel-Sets (mit dem Programm ip6tables) und der Bestimmung der Ein- und Ausgabe-Devices (siehe Firewalling/Netfilter6 für Details) vollzogen werden. In IPv4 ist das anders, forwarding per device ist hier möglich (hier wird am Interface, wo das Paket einlangt, die entsprechende Entscheidung getroffen).

Hiermit werden die Host/Router Einstellungen 'forwarding' aller Interfaces eingestellt (auch globales Forwarding genannt). Für weitere Details Siehe unten.

Ist der Wert gleich 0, dann ist IPv6 forwarding deaktiviert. Pakete verlassen in diesem Fall niemals ein anderes Interface (weder physikalische noch logische wie z.B. Tunnel).

11.2.3. conf/interface/*

Spezielle Einstellungen per Interface ändern.

Das funktionale Verhalten einzelner Einstellungen ist davon abhängig, ob lokales forwarding aktiviert ist oder nicht.

11.2.3.1. accept_ra

- Typ: BOOLEAN
- Standardeinstellung: aktiviert, wenn lokales forwarding deaktiviert ist. Deaktiviert, wenn lokales forwarding aktiviert ist.

Router Advertisements werden akzeptiert; das Interface wird mit Status 'received data' automatisch konfiguriert.

11.2.3.2. accept_redirects

- Typ: BOOLEAN
- Standardeinstellung: aktiviert, wenn lokales forwarding deaktiviert ist. Deaktiviert, wenn lokales forwarding aktiviert ist.

Akzeptiert von IPv6 Router gesendete Redirects.

11.2.3.3. autoconf

- Typ: BOOLEAN
- Funktionale Standardeinstellung: aktiviert, wenn accept_ra_pinfo aktiv ist. Deaktiviert, wenn accept_ra_pinfo deaktiviert ist.

Autokonfiguriert Adressen unter Benutzung der Prefix-Information eines Router-Advertisements.

11.2.3.4. dad_transmits

- Typ: INTEGER
- Standardwert: 1

Die Anzahl der gesendeten Proben zum entdecken von Adress-Duplikaten.

11.2.3.5. forwarding

- Typ: BOOLEAN
- Standardwert: FALSCH, wenn globale forwarding deaktiviert ist (Standard), ansonst WAHR

Konfigurieren von Interface-spezifischem Host/Router-Verhalten.

Anmerkung: Es wird die gleiche Konfiguration für alle Interfaces empfohlen; Gemischte Host/Router-Szenarios sind eher unüblich.

- Wert FALSCH: Per Standard wird von einem Host-Verhalten ausgegangen. Das bedeutet:

1. Der Schalter IsRouter ist bei Router Advertisements nicht aktiviert.
2. Router-Anfragen werden bei Bedarf gesendet.
3. Wenn accept_ra WAHR ist (Standard), dann werden Router Advertisements akzeptiert (und starte die automatische Konfiguration).
4. Wenn accept_redirects WAHR ist (Standard), dann akzeptiere Redirects.

- Wert WAHR: Ist lokales forwarding eingeschaltet, dann wird von einem Router-Verhalten ausgegangen. Das bedeutet genau das Gegenteil zu oben:

1. Der Schalter IsRouter ist bei Router Advertisements aktiviert.
2. Router-Anfragen werden nicht gesendet.
3. Router Advertisements werden ignoriert.
4. Redirects werden ignoriert.

11.2.3.6. hop_limit

- Typ: INTEGER
- Standardwert: 64

Der Standardwert für das Hop-Limit wird hiermit eingestellt.

11.2.3.7. mtu

- Type: INTEGER
- Standardwert: 1280 (Minimumwert in IPv6)

Der Standardwert für die Maximum Transfer Unit wird hiermit eingestellt.

11.2.3.8. router_solicitation_delay

- Typ: INTEGER
- Standardwert: 1

Die Anzahl der nach der Aktivierung eines Interfaces zu wartenden Sekunden bevor Router-Anfragen gesendet werden.

11.2.3.9. router_solicitation_interval

- Typ: INTEGER
- Standardwert: 4

Die Anzahl der Sekunden zwischen Router-Anfragen.

11.2.3.10. router_solicitations

- Typ: INTEGER
- Standardwert: 3

Die Anzahl der Router-Anfragen, bevor angenommen wird, dass keine Router verfügbar sind.

11.2.4. neigh/default/*

Standardeinstellungen der Neighbor-Erkennung und einige spezielle globale Intervall- sowie Threshold-Werte ändern:

11.2.4.1. gc_thresh1

- Typ: INTEGER
- Standardwert: 128

Mehr Infos hierzu in späteren Versionen.

11.2.4.2. gc_thresh2

- Typ: INTEGER
- Standardwert: 512

Mehr Infos hierzu in späteren Versionen.

11.2.4.3. gc_thresh3

- Typ: INTEGER
- Standardwert: 1024

Parameter zum Einstellen der Größe der Neighbour-Tabelle.

Wenn Sie viele Interfaces und Probleme mit inkorrekt oder nicht funktionierenden Routen haben, dann sollten Sie diesen Wert erhöhen. Ebenfalls erhöhen sollten Sie den Wert, wenn von einem aktiven Zebra (routing daemon) Folgendes angezeigt wird:

```
ZEBRA: netlink-listen error: No buffer space available, type=RTM_NEWROUTE(24),
¬ seq=426, pid=0
```

11.2.4.4. gc_interval

- Typ: INTEGER
- Standardwert: 30

Mehr Infos hierzu in späteren Versionen.

11.2.5. neigh/interface/*

Per Interface ändern spezieller Einstellungen zur Neighbor-Erkennung.

11.2.5.1. anycast_delay

- Typ: INTEGER
- Standardwert: 100

Mehr Infos hierzu in späteren Versionen.

11.2.5.2. gc_stale_time

- Typ: INTEGER
- Standardwert: 60

Mehr Infos hierzu in späteren Versionen.

11.2.5.3. proxy_qlen

- Typ: INTEGER
- Standardwert: 64

Mehr Infos hierzu in späteren Versionen.

11.2.5.4. unres_qlen

- Typ: INTEGER
- Standardwert: 3

Mehr Infos hierzu in späteren Versionen.

11.2.5.5. app_solicit

- Typ: INTEGER
- Standardwert: 0

Mehr Infos hierzu in späteren Versionen.

11.2.5.6. locktime

- Typ: INTEGER
- Standardwert: 0

Mehr Infos hierzu in späteren Versionen.

11.2.5.7. retrans_time

- Typ: INTEGER
- Standardwert: 100

Mehr Infos hierzu in späteren Versionen.

11.2.5.8. base_reachable_time

- Typ: INTEGER
- Standardwert: 30

Mehr Infos hierzu in späteren Versionen.

11.2.5.9. mcast_solicit

- Typ: INTEGER
- Standardwert: 3

Mehr Infos hierzu in späteren Versionen.

11.2.5.10. ucast_solicit

- Typ: INTEGER
- Standardwert: 3

Mehr Infos hierzu in späteren Versionen.

11.2.5.11. delay_first_probe_time

- Typ: INTEGER
- Standardwert: 5

Mehr Infos hierzu in späteren Versionen.

11.2.5.12. proxy_delay

- Typ: INTEGER
- Standardwert: 80

Mehr Infos hierzu in späteren Versionen.

11.2.6. route/*

Globale Routing-Einstellungen ändern.

11.2.6.1. flush

In neueren Kernel-Versionen wurde diese Option entfernt - mehr Infos hierzu in späteren Versionen.

11.2.6.2. gc_interval

- Typ: INTEGER
- Standardwert: 30

Mehr Infos hierzu in späteren Versionen.

11.2.6.3. gc_thresh

- Typ: INTEGER
- Standardwert: 1024

Mehr Infos hierzu in späteren Versionen.

11.2.6.4. mtu_expires

- Typ: INTEGER
- Standardwert: 600

Mehr Infos hierzu in späteren Versionen.

11.2.6.5. gc_elasticity

- Typ: INTEGER
- Standardwert: 0

Mehr Infos hierzu in späteren Versionen.

11.2.6.6. gc_min_interval

- Typ: INTEGER
- Standardwert: 5

Mehr Infos hierzu in späteren Versionen.

11.2.6.7. gc_timeout

- Typ: INTEGER
- Standardwert: 60

Mehr Infos hierzu in späteren Versionen.

11.2.6.8. min_adv_mss

- Typ: INTEGER
- Standardwert: 12

Mehr Infos hierzu in späteren Versionen.

11.2.6.9. max_size

- Typ: INTEGER
- Standardwert: 4096

Mehr Infos hierzu in späteren Versionen.

11.3. IPv6 relevante Einträge in /proc/sys/net/ipv4/

Zurzeit werden einige Schalter auch bei IPv6 eingesetzt (Dies bleibt so, bis IPv4 zur Gänze in ein unabhängiges Kernel-Modul umgewandelt wurde).

11.3.1. ip_*

11.3.1.1. ip_local_port_range

Diese Kontrolleinstellung wird ebenfalls bei IPv6 verwendet.

11.3.2. tcp_*

Diese Kontrolleinstellungen werden ebenfalls bei IPv6 verwendet.

11.3.3. icmp_*

Diese Kontrolleinstellungen werden bei IPv6 nicht verwendet. Zum aktivieren der ICMPv6 Quoten-Limitierung (auf Grund der ICMPv6 storms Auswirkungen sehr empfohlen) müssen netfilter-v6-Regeln eingesetzt werden.

11.3.4. Sonstige Einträge

Keine bekannt, bzw. von IPv6 vermutlich ungenutzt.

11.4. IPv6 relevante Einträge in /proc/net/

In /proc/net gibt es einige Einträge die ausschließlich Lese-Rechte besitzen. Mit "sysctl" können Sie hier keine Informationen bekommen, verwenden Sie anstelle dessen z.B. "cat".

11.4.1. if_inet6

- Typ: Eine Zeile pro Adresse mit jeweils mehreren Werten

Alle konfigurierten IPv6 Adressen werden hier in einem speziellen Format angezeigt. Im Beispiel wird ein Loopback-Interface angezeigt. Die Werte werden unten erklärt (siehe "net/ipv6/addrconf.c" für Details).

```
# cat /proc/net/if_inet6
00000000000000000000000000000001 01 80 10 80 lo
+--------------------------------+ ++ ++ ++ ++ ++
|                                  |  |  |  |  |
1                                  2  3  4  5  6
```

1. IPv6 Adresse mit 32 hexadezimalen Zeichen ohne Doppelpunkte als Trennzeichen
2. Netlink Device Nummer (Interface Index) im hexadezimalen Format (siehe auch "ip addr")
3. Präfix-Länge in hexadezimaler Notation
4. Wert des Gültigkeitsbereichs (s.a. Kernel Quellen "include/net/ipv6.h" und "net/ipv6/addrconf.c")
5. Interface flags (s.a. "include/linux/rtnetlink.h" und "net/ipv6/addrconf.c")
6. Devicename

11.4.2. ipv6_route

- Typ: Eine Zeile pro Route mit jeweils mehreren Werten

Alle konfigurierten IPv6 Routen werden hier in einem speziellen Format angezeigt. Im Beispiel wird ein Loopback-Interface angezeigt. Die Werte werden unten erklärt (siehe "net/ipv6/route.c" für Details).

```
# cat /proc/net/ipv6_route
00000000000000000000000000000000 00 00000000000000000000000000000000 00
+--------------------------------+ ++ +--------------------------------+ ++
|                                  |  |                                  |
1                                  2  3                                  4

¬ 00000000000000000000000000000000 ffffffff 00000001 00000001 00200200 lo
¬ +--------------------------------+ +------+ +------+ +------+ +------+ ++
¬ |                                  |        |        |        |        |
¬ 5                                  6        7        8        9        10
```

1. IPv6 Zielnetzwerk mit 32 hexadezimalen Zeichen ohne Doppelpunkte als Trennzeichen
2. IPv6 Ziel-Präfix-Länge in hexadezimaler Notation
3. IPv6 Ursprungsnetzwerk mit 32 hexadezimalen Zeichen ohne Doppelpunkte als Trennzeichen
4. IPv6 Ursprungs-Präfix-Länge in hexadezimaler Notation
5. IPv6 next Hop mit 32 hexadezimalen Zeichen ohne Doppelpunkte als Trennzeichen
6. Metrik in hexadezimaler Schreibweise
7. Reference Counter
8. Use Counter
9. Flags

10. Devicename

11.4.3. sockstat6

- Typ: Eine Zeile pro Protokoll mit Beschreibung und Wert

Statistiken über verwendete IPv6 Sockets. Beispiel:

```
# cat /proc/net/sockstat6
TCP6: inuse 7
UDP6: inuse 2
RAW6: inuse 1
FRAG6: inuse 0 memory 0
```

11.4.4. tcp6

Mehr Infos hierzu in späteren Versionen.

11.4.5. udp6

Mehr Infos hierzu in späteren Versionen.

11.4.6. igmp6

Mehr Infos hierzu in späteren Versionen.

11.4.7. raw6

Mehr Infos hierzu in späteren Versionen.

11.4.8. ip6_flowlabel

Mehr Infos hierzu in späteren Versionen.

11.4.9. rt6_stats

Mehr Infos hierzu in späteren Versionen.

11.4.10. snmp6

- Typ: Eine Zeile pro SNMP Beschreibung und Wert

SNMP Statistiken; diese können mittels snmp server und entsprechender MIB Tabelle mit einer Network Management Software gewonnen werden.

11.4.11. ip6_tables_names

Verfügbare netfilter6 Tabellen

Kapitel 12. Netlink-Interface zum Kernel

Mehr Infos hierzu in späteren Versionen... der Autor hat hiermit keine Erfahrung...

Kapitel 13. Adress-Auflösung

Die Auflösung von Namen zu einer IPv4- bzw. IPv6-Adresse wird üblicherweise durch die Benutzung einer libc resolver Bibliothek durchgeführt. Es sind einige Seltsamkeiten bekannt bei der Nutzung der Funktion *getaddrinfo*.

Mehr Information kann dazu aktuell gefunden werden unter Linux & IPv6: getaddrinfo and search domains - Research and RFC 3484 on Linux.

Mehr Infos hierzu in späteren Versionen...

Kapitel 14. Netzwerk-Fehlersuche

14.1. Server Socket-Anbindung

14.1.1. Überprüfung der Server Socket-Anbindung mit "netstat"

Es ist immer von Interesse welche Sockets eines Knotens gerade aktiv sind. Mit "netstat" können Sie die betreffenden Informationen abfragen:

Verwendete Optionen: -nlptu

Beispiel:

```
# netstat -nlptu
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
¬ PID/Program name
tcp        0      0 0.0.0.0:32768           0.0.0.0:*               LISTEN
¬ 1258/rpc.statd
tcp        0      0 0.0.0.0:32769           0.0.0.0:*               LISTEN
¬ 1502/rpc.mountd
tcp        0      0 0.0.0.0:515             0.0.0.0:*               LISTEN
¬ 22433/lpd Waiting
tcp        0      0 1.2.3.1:139             0.0.0.0:*               LISTEN
¬ 1746/smbd
tcp        0      0 0.0.0.0:111             0.0.0.0:*               LISTEN
¬ 1230/portmap
tcp        0      0 0.0.0.0:6000            0.0.0.0:*               LISTEN
¬ 3551/X
tcp        0      0 1.2.3.1:8081            0.0.0.0:*               LISTEN
¬ 18735/junkbuster
tcp        0      0 1.2.3.1:3128            0.0.0.0:*               LISTEN
¬ 18822/(squid)
tcp        0      0 127.0.0.1:953           0.0.0.0:*               LISTEN
¬ 30734/named
tcp        0      0 ::ffff:1.2.3.1:993      :::*                    LISTEN
¬ 6742/xinetd-ipv6
tcp        0      0 :::13                   :::*                    LISTEN
¬ 6742/xinetd-ipv6
tcp        0      0 ::ffff:1.2.3.1:143      :::*                    LISTEN
¬ 6742/xinetd-ipv6
tcp        0      0 :::53                   :::*                    LISTEN
¬ 30734/named
tcp        0      0 :::22                   :::*                    LISTEN
¬ 1410/sshd
tcp        0      0 :::6010                 :::*                    LISTEN
¬ 13237/sshd
udp        0      0 0.0.0.0:32768           0.0.0.0:*
¬ 1258/rpc.statd
udp        0      0 0.0.0.0:2049            0.0.0.0:*
¬ -
udp        0      0 0.0.0.0:32770           0.0.0.0:*
¬ 1502/rpc.mountd
udp        0      0 0.0.0.0:32771           0.0.0.0:*
¬ -
udp        0      0 1.2.3.1:137             0.0.0.0:*
¬ 1751/nmbd
udp        0      0 0.0.0.0:137             0.0.0.0:*
```

```
¬ 1751/nmbd
udp        0      0 1.2.3.1:138             0.0.0.0:*
¬ 1751/nmbd
udp        0      0 0.0.0.0:138             0.0.0.0:*
¬ 1751/nmbd
udp        0      0 0.0.0.0:33044           0.0.0.0:*
¬ 30734/named
udp        0      0 1.2.3.1:53              0.0.0.0:*
¬ 30734/named
udp        0      0 127.0.0.1:53            0.0.0.0:*
¬ 30734/named
udp        0      0 0.0.0.0:67              0.0.0.0:*
¬ 1530/dhcpd
udp        0      0 0.0.0.0:67              0.0.0.0:*
¬ 1530/dhcpd
udp        0      0 0.0.0.0:32858           0.0.0.0:*
¬ 18822/(squid)
udp        0      0 0.0.0.0:4827            0.0.0.0:*
¬ 18822/(squid)
udp        0      0 0.0.0.0:111             0.0.0.0:*
¬ 1230/portmap
udp        0      0 :::53                   :::*
¬ 30734/named
```

14.2. tcpdump-Beispiele

Hier folgen einige Beispiele von (mit tcpdump) aufgezeichneten Paketen, die hoffentlich bei Ihrer Fehlersuche nützlich sein können.

... mehr Beispiele in den nächsten Versionen...

14.2.1. Router-Erkennung

14.2.1.1. Router Advertisement

```
15:43:49.484751 fe80::212:34ff:fe12:3450 > ff02::1: icmp6: router
¬ advertisement(chlim=64, router_ltime=30, reachable_time=0,
¬ retrans_time=0)(prefix info: AR valid_ltime=30, preffered_ltime=20,
¬ prefix=2002:0102:0304:1::/64)(prefix info: LAR valid_ltime=2592000,
¬ preffered_ltime=604800, prefix=2001:0db8:0:1::/64)(src lladdr:
¬ 0:12:34:12:34:50) (len 88, hlim 255)
```

Der Router mit der link-lokalen Adresse "fe80::212:34ff:fe12:3450" sendet eine Ankündigung mit zwei Präfixes "2002:0102:0304:1::/64" (Lebensdauer 30s) und "2001:0db8:0:1::/64" (Lebensdauer 2592000s) sowie der eigenen Schicht 2 MAC Adresse "0:12:34:12:34:50" an die all-node-on-link Multicast Adresse "ff02::1".

14.2.1.2. Router Anfrage

```
15:44:21.152646 fe80::212:34ff:fe12:3456 > ff02::2: icmp6: router solicitation
¬ (src lladdr: 0:12:34:12:34:56) (len 16, hlim 255)
```

Der Knoten mit der link-lokalen Adresse "fe80:212:34ff:fe12:3456" und der Schicht 2 MAC Adresse "0:12:34:12:34:56" sucht nach einem Router und sendet hierfür diese Anfrage an die all-router-on-link Multicast Adresse "ff02::2".

14.2.2. Neighbor-Erkennung

14.2.2.1. Neighbor discovery solicitation zur Entdeckung doppelter Adressen

Folgende Pakete werden vom Knoten mit der Schicht 2 MAC Adresse "0:12:34:12:34:56" während der automatischen Konfiguration an die solicited-node link-lokale Multicast Adresse gesendet. Es wird überprüft, ob eine potentielle Adresse bereits von einem anderen Knoten am Link verwendet wird.

- Der Knoten will seine link-lokale Adresse "fe80:212:34ff:fe12:3456" konfigurieren und überprüft auf Duplikate

```
15:44:17.712338 :: > ff02::1:ff12:3456: icmp6: neighbor sol: who has
¬ fe80::212:34ff:fe12:3456(src lladdr: 0:12:34:12:34:56) (len 32, hlim 255)
```

- Der Knoten will seine globale Adresse "2002:0102:0304:1:212:34ff:fe12:3456" konfigurieren (nach Empfang eines Advertisements wie weiter oben abgebildet) und überprüft auf Duplikate

```
15:44:21.905596 :: > ff02::1:ff12:3456: icmp6: neighbor sol: who has
¬ 2002:0102:0304:1:212:34ff:fe12:3456(src lladdr: 0:12:34:12:34:56) (len 32,
¬ hlim 255)
```

- Der Knoten will seine globale Adresse "2001:0db8:0:1:212:34ff:fe12:3456" konfigurieren (nach Empfang eines Advertisements wie weiter oben abgebildet) und überprüft auf Duplikate

```
15:44:22.304028 :: > ff02::1:ff12:3456: icmp6: neighbor sol: who has
¬ 2001:0db8:0:1:212:34ff:fe12:3456(src lladdr: 0:12:34:12:34:56) (len 32, hlim
¬ 255)
```

14.2.2.2. Neighbor discovery solicitation zur Host oder Gateway-Suche

- Der Knoten möchte Pakete an die Adresse "2001:0db8:0:1::10" senden, hat hierfür aber keine Schicht 2 MAC Adresse und sendet aus diesem Grund zuerst eine Anfrage

```
13:07:47.664538 2002:0102:0304:1:2e0:18ff:fe90:9205 > ff02::1:ff00:10: icmp6:
¬ neighbor sol: who has 2001:0db8:0:1::10(src lladdr: 0:e0:18:90:92:5) (len 32,
¬ hlim 255)
```

- Der Knoten sucht nun nach der Adresse "fe80::10"

```
13:11:20.870070 fe80::2e0:18ff:fe90:9205 > ff02::1:ff00:10: icmp6: neighbor
¬ sol: who has fe80::10(src lladdr: 0:e0:18:90:92:5) (len 32, hlim 255)
```

Kapitel 15. Unterstützung einer ständigen IPv6-Konfiguration in Linux Distributionen

Einige Linux-Distributionen unterstützen bereits eine permanente IPv6 Konfiguration. Hierbei werden sowohl bestehende oder als auch neue Konfiguration- und Skriptdateien verwendet sowie tlw. IPv4 Skripte abgeändert.

15.1. Red Hat Linux und "Klone"

Seitdem der Autor begann das IPv6 & Linux - HowTo zu schreiben, war es seine Absicht eine permanente IPv6 Konfiguration zu ermöglichen, wobei die gebräuchlichsten Anwendungsszenarien wie Host-only, Router-only, Dual-homed-host, Router mit einem zweiten Netzwerkstrang, normale Tunnel, 6to4 Tunnel, etc. abgedeckt sein sollten. Heute gibt es eine Sammlung von Konfigurations- und Skriptdateien, die genau diesem Zweck erfüllen (es wurden nie echte Probleme gemeldet, allerdings ist unbekannt von wie vielen Personen dieses Set benutzt wird). Diese Dateien werden von Zeit zu Zeit erweitert und es gibt inzwischen eine eigene Homepage hierfür: initscripts-ipv6 homepage (Spiegel). Da der Autor seine ersten Schritte mit IPv6 auf einem Red Hat Linux 5.0 Klon gemacht habe, basieren seine IPv6 Entwicklungssysteme heute zumeist auf Red Hat Linux und die Skriptdateien sind folglich primär für diese Distributionen gedacht. Es war ebenfalls sehr einfach bestehende Konfigurationsdateien zu erweitern, neue zu erstellen und den Start des IPv6 Setup in das IPv4 Setup einzubetten.

Erfreulicherweise beinhaltet Red Hat Linux seit der Version 7.1 die IPv6-Skripts des Autors. Unterstützt wurde dies und wird auch weiterhin von Pekka Savola.

Bei Mandrake ist ab Version 8.0 ebenfalls ein IPv6-fähiges initscript Paket beinhaltet, ein kleiner Fehler verhindert aber nach wie vor die Anwendung ("ifconfig" vermisst "inet6" vor "add").

15.1.1. Test der IPv6-Unterstützung bei Netzwerk-Konfigurations-Scripts

Sie können überprüfen, ob Ihre Distribution eine permanente IPv6 Konfiguration unter Verwendung der Skript-Sammlung des Autors unterstützt. Folgende script library sollte existieren:

```
/etc/sysconfig/network-scripts/network-functions-ipv6
```

Automatischer Test:

```
# test -f /etc/sysconfig/network-scripts/network-functions-ipv6 && echo "Main
¬ IPv6 script library exists"
```

Die Versionsnummer der Library ist von Interesse, wenn Sie Features vermissen sollten. Die Versionsnummer können Sie anzeigen, indem Sie folgenden Befehl ausführen (einfacher ist es sicherlich, wenn Sie im Header der Datei nachlesen):

```
# source /etc/sysconfig/network-scripts/network-functions-ipv6 &&
¬ getversion_ipv6_functions
20011124
```

Im obigen Beispiel ist die Versionsnummer 20011124. Um zu sehen, was sich inzwischen geändert hat, können Sie hier die neuesten Informationen nachlesen: initscripts-ipv6 homepage (Spiegel). Sie finden hier auch ein Change-Log.

15.1.2. Kurze Anleitung zum aktivieren von IPv6 bei RHL 7.1, 7.2, 7.3, ...

- Überprüfen Sie, ob das IPv6 Modul auf Ihrem System bereits geladen ist:

```
# modprobe -c | grep net-pf-10
alias net-pf-10 off
```

- Ist das Ergebnis "off", dann aktivieren Sie IPv6 durch hinzufügen folgender Zeile in /etc/sysconfig/network

```
NETWORKING_IPV6=yes
```

- Rebooten bzw. starten Sie das Netzwerk neu mit dem Befehl

```
# service network restart
```

- Nun sollte das IPv6 Modul geladen sein

```
# modprobe -c | grep ipv6
alias net-pf-10 ipv6
```

Ist ihr System an einem Link, der Router Advertisements liefert, dann wird die automatische Konfiguration automatisch durchgeführt. Zusätzlich Informationen darüber, welche Einstellungen unterstützt werden finden Sie in der Datei /usr/share/doc/initscripts-$version/sysconfig.txt.

15.2. SuSE Linux

Seit neueren 7.x Versionen gibt es eine wirklich rudimentäre Unterstützung für IPv6, siehe /etc/rc.config für Details.

Aufgrund der komplett unterschiedlichen Struktur der Konfigurations- und Scriptdateien ist es sehr schwer (oder unmöglich) das Set für Red Hat Linux und seine Klone mit dieser Distribution zu verwenden. In Version 8.x wurde das Konfigurations-Setup bei SuSE komplett abgeändert.

15.2.1. SuSE Linux 7.3

- How to setup 6to4 IPv6 with SuSE 7.3

15.2.2. SuSE Linux 8.0

15.2.2.1. IPv6-Adress-Konfiguration

Editiere Datei /etc/sysconfig/network/ifcfg-<Interface-Name> und setze folgende Variable entsprechend

```
IP6ADDR="<ipv6-Adresse>/<prefix>"
```

15.2.2.2. Zusätzliche information

Siehe Datei /usr/share/doc/packages/sysconfig/README

15.2.3. SuSE Linux 8.1

15.2.3.1. IPv6-Adress-Konfiguration

Editiere Datei /etc/sysconfig/network/ifcfg-<Interface-Name> und setze folgende Variable entsprechend

```
IPADDR="<ipv6-Adresse>/<prefix>"
```

15.2.3.2. Zusätzliche information

Siehe Datei /usr/share/doc/packages/sysconfig/Network

15.3. Debian Linux

Folgende Information wurde von Stephane Bortzmeyer <bortzmeyer at nic dot fr> beigesteuert.

1. Überprüfe, ob IPv6 aktiv ist, entweder weil es in den Kernel hineinkompilier oder das Modul geladen wurde. Für die letzte Möglichkeit gibt es 3 Lösungen: Editieren der Datei /etc/modules, Benutzung des Features pre-up (siehe unten) oder Benutzung von kmod (wird hier nicht weiter erklärt).
2. Konfiguriere die Schnittstelle (hier im Beispiel: eth0). Editiere /etc/network/interfaces :

```
iface eth0 inet6 static
        pre-up modprobe ipv6
        address 2001:0db8:1234:5::1:1
        # To suppress completely autoconfiguration:
        # up echo 0 > /proc/sys/net/ipv6/conf/all/autoconf
        netmask 64
        # The router is autoconfigured and has no fixed address.
        # It is magically
        # found. (/proc/sys/net/ipv6/conf/all/accept_ra). Otherwise:
        #gateway 2001:0db8:1234:5::1
```

Danach rebooten oder folgendes Kommando ausführen

```
# ifup --force eth0
```

Danach sollte die statische IPv6-Adresse konfiguriert sein.

15.3.1. Weiterführende Informationen

- IPv6 with Debian Linux
- Jean-Marc V. Liotier's HOWTO for Freenet6 & Debian Users (am 24.12.2002 in der mailinglist users@ipv6.org angekündigt)

Kapitel 16. Automatische Konfiguration

16.1. Stateless Auto-Konfiguration

Wird unterstützt und kann bei der zugewiesenen link-lokalen Adressen beobachtet werden, sobald ein IPv6 fähiges Interface aktiv ist.

Beispiel:

```
# ip -6 addr show dev eth0 scope link
2: eth0: <BROADCAST,MULTICAST,UP> mtu 1500 qlen1000
    inet6 fe80::211:d8ff:fe6b:f0f5/64 scope link
       valid_lft forever preferred_lft forever
```

16.2. Stateful Auto-Konfiguration unter Verwendung des Router Advertisement Daemon (radvd)

Mehr Infos hierzu in späteren Versionen. Siehe unten im Abschnitt radvd daemon autoconfiguration.

16.3. Dynamic Host Configuration Protocol v6 (DHCPv6)

Nach einer langen Zeit der Diskussion wurde RFC 3315 / Dynamic Host Configuration Protocol for IPv6 (DHCPv6) verabschiedet. Momentan (10/2005) existieren 2 Implementierungen:

- Dibbler von Tomasz Mrugalski <thomson at klub dot com dot pl>
- DHCPv6 on Sourceforge (Tipps zur Konfiguration)

Kapitel 17. Mobilität

17.1. Allgemeines

17.1.1. Mobilität eines Knotens (Node Mobility)

Die Unterstützung für IPv6-Mobilität in Linux kann durch die Installation der MIPL2-Implementierung aktviert werden, welche hier zu finden ist: http://www.mobile-ipv6.org/

Diese Implementierung ist konform zur RFC 3775. Sie besteht aus einem Kernel-Patch und einen Mobilitäts-Daemon (genannt mip6d). Die Version 2.0.1 passt für Linux kernel 2.6.15.

Installation und Setup sind im Linux Mobile IPv6 HOWTO beschrieben.

17.1.2. Netzwerk-Mobililtät

Zusätzlich existiert die Implementierung der Netzwerk-Mobilität für Linux, genannt NEPL, und basiert auf MIPL. Diese steht auch zur Verfügung unter: http://www.mobile-ipv6.org/.

Folgendes HOWTO Dokument beschreibt Setup und Konfiguration:
http://www.nautilus6.org/doc/nepl-howto/.

17.1.3. Links

- Mobile IPv6 for Linux (MIPL) project: http://www.mobile-ipv6.org/
- Nautilus6 working group: http://nautilus6.org/
- Fast Handovers for Mobile IPv6 for Linux project: http://www.fmipv6.org/
- USAGI-patched Mobile IPv6 for Linux (UMIP):http://umip.linux-ipv6.org/
- Deploying IPsec/IKE-protected MIPv6 under Linux:http://natisbad.org/MIPv6/
- RFC 3775 / Mobility Support in IPv6
- RFC 3776 / Using IPsec to Protect Mobile IPv6 Signaling Between Mobile Nodes and Home Agents
- RFC 3963 / Network Mobility (NEMO)
- RFC 4068 / Fast Handovers for Mobile IPv6
- RFC 4423 / Host Identity Protocol (HIP) Architecture
- RFC 5201 / Host Identity Protocol
- HIP Implementierungen: http://infrahip.hiit.fi/, http://hip4inter.net/, http://www.openhip.org/

Kapitel 18. Firewall-Funktionalität

Die IPv6 Firewall-Funktionalität ist wichtig; vor allem dann, wenn Sie auf Ihren internen Netzen IPv6 mit globalen IPv6 Adressen einsetzen. In IPv6 werden - im Unterschied zu IPv4, wo interne Hosts automatisch durch private IPv6 Adressen geschützt werden (RFC 1918 / Address Allocation for Private Internets bzw. Google search for Microsoft + APIPA) - globale Adressen verwendet und jeder mit IPv6-Anbindung kann alle internen Knoten, bei denen IPv6 aktiv ist, erreichen.

18.1. Firewall-Funktionalität mit netfilter6

Von Haus aus unterstützt wird die IPv6-Firewall-Funktionalität im Kernel erst ab Version 2.4+. In älteren 2.2+ Versionen können sie nur mit Protocol 41 das generelle Tunnel von IPv6-in-IPv4-Paketen filtern.

Achtung: Es gibt keine Garantie, dass die beschriebenen Regeln und Beispiele ihr System auch wirklich schützen können!

Beobachten Sie nach der Installation ihr Regelset, siehe Abschnitt Abschnitt 19.3.

Kernels ab Version 2.6.20 unterstützen den IPv6-Verbindungsstatus (connection tracking) vollständig.

18.1.1. Weitere Informationen

- Netfilter project
- maillist archive of netfilter users
- maillist archive of netfilter developers
- Unofficial status informations

18.2. Vorbereitung

Dies ist nur notwendig, wenn der mitgelieferte Kernel und Netfilter nicht den Ansprüchen genügt und neue Featurs bereits verfügbar sind, jedoch noch nicht beinhaltet.

18.2.1. Quellen besorgen

Besorgen Sie sich den aktuellsten Kernel: http://www.kernel.org/

Besorgen Sie sich das aktuellste iptables Paket:

- Source tarball (für Kernel Patches): http://www.netfilter.org/

18.2.2. Quellen entpacken

Wechseln Sie in das Source-Verzeichnis:

```
# cd /path/to/src
```

Entpacken sie die Kernel-Quellen und vergeben diesen einen neuen Namen

```
# tar z|jxf kernel-version.tar.gz|bz2
# mv linux linux-version-iptables-version+IPv6
```

Entpacken Sie die iptables Quellen

```
# tar z|jxf iptables-version.tar.gz|bz2
```

18.2.3. Neueste iptables/IPv6-relevante Patches den Kernel-Quellen hinzufügen

Wechseln Sie in das iptables Verzeichnis

```
# cd iptables-version
```

Fügen Sie relevante Patches hinzu

```
# make pending-patches KERNEL_DIR=/path/to/src/linux-version-iptables-version/
```

Fügen Sie zusätzliche IPv6 relevante IPv6 Patches hinzu (die nach wie vor nicht im Standard-Kernel enthalten sind)

```
# make patch-o-matic KERNEL_DIR=/path/to/src/linux-version-iptables-version/
```

Sagen Sie zu folgenden Optionen (iptables-1.2.2) Ja:

- ah-esp.patch
- masq-dynaddr.patch (nur benötigt bei Systemen mit dynamischer IP-Zuweisung am WAN mittels PPP oder PPPoE)
- ipv6-agr.patch.ipv6
- ipv6-ports.patch.ipv6
- LOG.patch.ipv6
- REJECT.patch.ipv6

Überprüfen Sie die Erweiterungen

```
# make print-extensions
Extensions found: IPv6:owner IPv6:limit IPv6:mac IPv6:multiport
```

18.2.4. Konfiguration, kompilieren und Installation eines neues Kernels

Wechseln Sie zu den Kernel-Quellen

```
# cd /path/to/src/linux-version-iptables-version/
```

Editieren Sie das Makefile

```
- EXTRAVERSION =
+ EXTRAVERSION = -iptables-version+IPv6-try
```

Starten Sie configure und aktivieren Sie IPv6 relevante Optionen

```
           Code maturity level options
                 Prompt for development and/or incomplete code/drivers : yes
           Networking options
                 Network packet filtering: yes
                 The IPv6 protocol: module
                      IPv6: Netfilter Configuration
                            IP6 tables support: module
                            All new options like following:
                                  limit match support: module
                                  MAC address match support: module
```

```
                                    Multiple port match support: module
                                    Owner match support: module
                                    netfilter MARK match support: module
                                    Aggregated address check: module
                                    Packet filtering: module
                                         REJECT target support: module
                                         LOG target support: module
                                    Packet mangling: module
                                    MARK target support: module
```

Konfigurieren Sie bei Bedarf Sonstiges abseits von IPv6.

Kompilieren und Installation: siehe Kapitel Kernel sowie andere HOWTOs.

18.2.5. iptables neu kompilieren und installieren

Stellen Sie sicher, dass obige Kernel-Sourceverzeichnisstruktur unter /usr/src/linux liegt

Benennen sie das ältere Verzeichnis um

```
# mv /usr/src/linux /usr/src/linux.old
```

Erstellen Sie einen neuen symbolischen Link

```
# ln -s /path/to/src/linux-version-iptables-version /usr/src/linux
```

Erstellen Sie ein neues SRPMS

```
# rpm --rebuild /path/to/SRPMS/iptables-version-release.src.rpm
```

Installieren Sie das neue iptables Paket (iptables + iptables-ipv6)

- Bei RH 7.1 Systemen ist normalerweise eine ältere Version hiervon bereits installiert, verwenden Sie daher die Option "Freshen":

```
# rpm -Fhv /path/to/RPMS/cpu/iptables*-version-release.cpu.rpm
```

- Ist keine ältere Version installiert, benutzen Sie die Option "install":

```
# rpm -ihv /path/to/RPMS/cpu/iptables*-version-release.cpu.rpm
```

- Bei RH 6.2 Systemen ist normalerweise kein Kernel Version 2.4.x installiert und die Anforderungen sind demnach nicht gegeben. Benutzen Sie in diesem Fall "nodeps":

```
# rpm -ihv --nodeps /path/to/RPMS/cpu/iptables*-version-release.cpu.rpm
```

Damit iptables die Libraries finden kann, ist es eventuell notwendig, einen symbolischen Link für die iptables Libraries zu erstellen:

```
# ln -s /lib/iptables/ /usr/lib/iptables
```

18.3. Verwendung

18.3.1. Unterstützung im Kernel

Laden Sie das Modul (falls dies im Kernel so kompiliert wurde):

```
# modprobe ip6_tables
```

Überprüfen der IPv6-Unterstützung:

```
# [ ! -f /proc/net/ip6_tables_names ] && echo "Current kernel doesn't support
¬ 'ip6tables' firewalling (IPv6)!"
```

18.3.2. Die Benützung von iptables lernen

18.3.2.1. Auflistung aller netfilter Einträge

- Kurze Auflistung:

```
# ip6tables -L
```

- Erweiterte Auflistung:

```
# ip6tables -n -v --line-numbers -L
```

18.3.2.2. Auflistung angegebener Filter

```
# ip6tables -n -v --line-numbers -L INPUT
```

18.3.2.3. Hinzufügen einer Log-Regel zum Input-Filter mit Optionen

```
# ip6tables --table filter --append INPUT  -j LOG --log-prefix "INPUT:"
¬ --log-level 7
```

18.3.2.4. Hinzufügen einer Drop-Regel zum Input-Filter

```
# ip6tables --table filter --append INPUT  -j DROP
```

18.3.2.5. Löschen einer Regel mit Hilfe der Regelnummer

```
# ip6tables --table filter --delete INPUT 1
```

18.3.2.6. Aktiviere die Auswertung des Verbindungsstatus (connection tracking)

Seit Kernel-Version 2.6.20 ist die Auswertung des IPv6-Verbindungsstatus gut unterstützt. Die bis dahin statuslosen Filterregeln sollten ersetzt werden..

```
# ip6tables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
```

18.3.2.7. ICMPv6 erlauben

Bei älteren Kernelversionen (unpatched kernel 2.4.5 und iptables-1.2.2) kann keine nähere Spezifizierung des ICMPv6-Typs vorgenommen werden:

- Eingehender ICMPv6 Verkehr durch Tunnel erlauben

```
# ip6tables -A INPUT -i sit+ -p icmpv6 -j ACCEPT
```

- Ausgehenden ICMPv6 Verkehr durch Tunnel erlauben

```
# ip6tables -A OUTPUT -o sit+ -p icmpv6 -j ACCEPT
```

Neuere Kernel erlauben das Spezifizieren des ICMPv6-Typs:

```
# ip6tables -A INPUT -p icmpv6 --icmpv6-type echo-request -j ACCEPT
```

18.3.2.8. Rate-limiting

Da es zu einem ICMPv6 Storm kommen kann (der Autor hat dies bereits mehrfach beobachtet), sollten sie das rate limiting zumindest für das ICMP Regelset einsetzen. Zusätzlich sollten auch die Logging Regeln mit rate limiting geschützt werden, um DoS Attacken gegen das syslog sowie gegen die Logdateien enthaltenden Patitionen entgegenzuwirken. Ein Beispiel für ein rate limited ICMPv6 sieht wie folgt aus:

```
# ip6tables -A INPUT --protocol icmpv6 --icmpv6-type echo-request
¬ -j ACCEPT --match limit --limit 30/minute
```

18.3.2.9. Eingehende SSH-Verbindung erlauben

Im folgenden Beispiel werden eingehende SSH-Verbindungen von einer speziellen IPv6 Adresse zugelassen:

- Eingehende SSH Verbindungen werden von der Adresse 2001:0db8:100::1/128 erlaubt

```
# ip6tables -A INPUT -i sit+ -p tcp -s 2001:0db8:100::1/128 --sport 512:65535
¬ --dport 22 -j ACCEPT
```

- Erlaube Antwortpakete (nicht mehr notwendig, wenn der IPv6-Verbindungsstatus ausgewertet wird!)

```
# ip6tables -A OUTPUT -o sit+ -p tcp -d 2001:0db8:100::1/128 --dport 512:65535
¬ --sport 22 ! --syn -j ACCEPT
```

18.3.2.10. Getunnelten IPv6-in-IPv4 Datenverkehr erlauben

Um getunnelte IPv6-in-IPv4 Pakete zu akzeptieren, müssen Sie in Ihrem IPv4 Firewall-Setup entsprechende Regeln einzufügen, z.B.

- Akzeptiere eingehende IPv6-in-IPv4 Daten am interface ppp0

```
# iptables -A INPUT -i ppp0 -p ipv6 -j ACCEPT
```

- Akzeptiere ausgehende IPv6-in-IPv4 Daten am interface ppp0

```
# iptables -A OUTPUT -o ppp0 -p ipv6 -j ACCEPT
```

Haben Sie nur einen statischen Tunnel, dann können sie die IPv4 Adresse auch dediziert angeben:

- Akzeptiere eingehende IPv6-in-IPv4 Daten vom Tunnel-Endpunkt 192.0.2.2. am interface ppp0

```
# iptables -A INPUT -i ppp0 -p ipv6 -s 192.0.2.2 -j ACCEPT
```

- Akzeptiere ausgehende IPv6-in-IPv4 Daten vom Tunnel-Endpunkt 1.2.3.4 am interface ppp0

```
# iptables -A OUTPUT -o ppp0 -p ipv6 -d 192.0.2.2 -j ACCEPT
```

18.3.2.11. Schutz gegen eingehende TCP-Verbindungs-Anfragen

SEHR EMPFOHLEN! Aus Sicherheitsgründen sollten Sie auf jeden Fall eine Regel inkludieren, wodurch eingehende TCP-Verbindungs-Anfragen geblockt werden. Wenn Sie andere Interfacenamen verwenden, müssen Sie die Option "-i" entsprechend anpassen!

- Blockiere eingehende TCP-Verbindungs-Anfragen zu diesem Host

```
# ip6tables -I INPUT -i sit+ -p tcp --syn -j DROP
```

- Blockiere eingehende TCP-Verbindungs-Anfragen zu Hosts hinter diesem Router

```
# ip6tables -I FORWARD -i sit+ -p tcp --syn -j DROP
```

Eventuell müssen diese Regeln unterhalb anderer Regeln platziert werden. Nehmen Sie sich für die Reihenfolge der Regeln etwas Zeit. Sinnvoll wird es auch sein, ein Script mit den Regeln zu erstellen, damit die Regeln in der gewünschten Reihenfolge angewendet werden.

18.3.2.12. Schutz gegen eingehende UDP-Verbindungs-Anfragen

EBENFALLS SEHR EMPHOLEN! Wie bereits im Kapitel Firewall erwähnt, ist es möglich die Ports bei ausgehenden UDP/TCP-Verbindungen zu kontrollieren. Im Falle, dass all Ihre IPv6 Systeme lokale Ports verwenden, z.B. von 32768 bis 60999, dann können sie ebenfalls UDP Verbindungen filtern (bis das Verbindungs-Tracking funktioniert):

- Blockiere eingehende UDP-Pakete, die nicht Antworten ausgehender Anfragen dieses Host sein können

```
# ip6tables -I INPUT -i sit+ -p udp ! --dport 32768:60999 -j DROP
```

- Blockiere eingehende UDP-Pakete, die nicht Antworten auf Anfragen von hinter diesem Router gelegenen Hosts sein können

```
# ip6tables -I FORWARD -i sit+ -p udp ! --dport 32768:60999 -j DROP
```

18.3.3. Anwendungsbeispiele

18.3.3.1. Einfaches Beispiel für Fedora

Folgende Zeilen zeigen eine einfache Firewall-Konfiguration für Fedora 6 (ab Kernel-Version 2.6.20). Ausgehend von dem Origina (generiert durch system-config-firewall) wurden Modifikationen für die Unterstützung des Verbindungsstatus und der Rückgabe der passenden ICMPv6-Meldung für Rejects. Eingehende SSH (Port 22) Verbindungen sind erlaubt.

```
Datei: /etc/sysconfig/ip6tables
```

```
*filter :INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
:RH-Firewall-1-INPUT - [0:0]
-A INPUT -j RH-Firewall-1-INPUT
-A FORWARD -j RH-Firewall-1-INPUT
-A RH-Firewall-1-INPUT -i lo -j ACCEPT
-A RH-Firewall-1-INPUT -p icmpv6 -j ACCEPT
-A RH-Firewall-1-INPUT -p 50 -j ACCEPT
-A RH-Firewall-1-INPUT -p 51 -j ACCEPT
-A RH-Firewall-1-INPUT -p udp --dport 5353 -d ff02::fb -j ACCEPT
-A RH-Firewall-1-INPUT -p udp -m udp --dport 631 -j ACCEPT
-A RH-Firewall-1-INPUT -p tcp -m tcp --dport 631 -j ACCEPT
-A RH-Firewall-1-INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
-A RH-Firewall-1-INPUT -m state --state NEW -p tcp --dport 22 -j ACCEPT
-A RH-Firewall-1-INPUT -j REJECT --reject-with icmp6-adm-prohibited
COMMIT
```

Zwecks der Vollständigkeit ist hier auch die entsprechende Konfiguration für IPv4 gezeigt:

```
Datei: /etc/sysconfig/iptables

*filter :INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
:RH-Firewall-1-INPUT - [0:0]
-A INPUT -j RH-Firewall-1-INPUT
-A FORWARD -j RH-Firewall-1-INPUT
-A RH-Firewall-1-INPUT -i lo -j ACCEPT
-A RH-Firewall-1-INPUT -p icmp --icmp-type any -j ACCEPT
-A RH-Firewall-1-INPUT -p 50 -j ACCEPT
-A RH-Firewall-1-INPUT -p 51 -j ACCEPT
-A RH-Firewall-1-INPUT -p udp --dport 5353 -d 224.0.0.251 -j ACCEPT
-A RH-Firewall-1-INPUT -p udp -m udp --dport 631 -j ACCEPT
-A RH-Firewall-1-INPUT -p tcp -m tcp --dport 631 -j ACCEPT
-A RH-Firewall-1-INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
-A RH-Firewall-1-INPUT -m state --state NEW -m tcp -p tcp --dport 22 -j ACCEPT
-A RH-Firewall-1-INPUT -j REJECT --reject-with icmp-host-prohibited
COMMIT
```

Benutzung:

- Erzeugen/Modifizieren der Konfigurationsdateien
- Aktivieren von IPv4 & IPv6 Firewalling

```
# service iptables start
# service ip6tables start
```

- Aktivieren des automatischen Starts nach dem Reboot

```
# chkconfig iptables on
# chkconfig ip6tables on
```

18.3.3.2. Umfangreicheres Beispiel

Folgende Zeilen zeigen ein umfangreicheres Setup. Happy netfilter6 Regelset erstellen...

```
# ip6tables -n -v -L
Chain INPUT (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
```

```
    0     0 extIN      all      sit+   *       ::/0                ::/0
    4   384 intIN      all      eth0   *       ::/0                ::/0
    0     0 ACCEPT     all      *      *       ::1/128             ::1/128
    0     0 ACCEPT     all      lo     *       ::/0                ::/0
    0     0 LOG        all      *      *       ::/0                ::/0
¬        LOG flags 0 level 7 prefix `INPUT-default:'
    0     0 DROP       all      *      *       ::/0                ::/0

Chain FORWARD (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source              destination
¬
    0     0 int2ext    all      eth0   sit+    ::/0                ::/0
    0     0 ext2int    all      sit+   eth0    ::/0                ::/0
    0     0 LOG        all      *      *       ::/0                ::/0
¬        LOG flags 0 level 7 prefix `FORWARD-default:'
    0     0 DROP       all      *      *       ::/0                ::/0

Chain OUTPUT (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source              destination
¬
    0     0 extOUT     all      *      sit+    ::/0                ::/0
    4   384 intOUT     all      *      eth0    ::/0                ::/0
    0     0 ACCEPT     all      *      *       ::1/128             ::1/128
    0     0 ACCEPT     all      *      lo      ::/0                ::/0
    0     0 LOG        all      *      *       ::/0                ::/0
¬        LOG flags 0 level 7 prefix `OUTPUT-default:'
    0     0 DROP       all      *      *       ::/0                ::/0

Chain ext2int (1 references)
 pkts bytes target     prot opt in     out     source              destination
¬
    0     0 ACCEPT     icmpv6    *      *       ::/0                 ::/0
    0     0 ACCEPT     tcp      *      *       ::/0                ::/0
¬        tcp spts:1:65535 dpts:1024:65535 flags:!0x16/0x02
    0     0 LOG        all      *      *       ::/0                ::/0
¬        LOG flags 0 level 7 prefix `ext2int-default:'
    0     0 DROP       tcp      *      *       ::/0                ::/0
    0     0 DROP       udp      *      *       ::/0                ::/0
    0     0 DROP       all      *      *       ::/0                ::/0

Chain extIN (1 references)
 pkts bytes target     prot opt in     out     source              destination
¬
    0     0 ACCEPT     tcp      *      *       3ffe:400:100::1/128  ::/0
¬        tcp spts:512:65535 dpt:22
    0     0 ACCEPT     tcp      *      *       3ffe:400:100::2/128  ::/0
¬        tcp spts:512:65535 dpt:22
    0     0 ACCEPT     icmpv6    *      *       ::/0                 ::/0
    0     0 ACCEPT     tcp      *      *       ::/0                ::/0
¬        tcp spts:1:65535 dpts:1024:65535 flags:!0x16/0x02
    0     0 ACCEPT     udp      *      *       ::/0                ::/0
¬        udp spts:1:65535 dpts:1024:65535
    0     0 LOG        all      *      *       ::/0                ::/0
¬        limit: avg 5/min burst 5 LOG flags 0 level 7 prefix `extIN-default:'
    0     0 DROP       all      *      *       ::/0                ::/0

Chain extOUT (1 references)
 pkts bytes target     prot opt in     out     source              destination
¬
    0     0 ACCEPT     tcp      *      *       ::/0
¬ 2001:0db8:100::1/128tcp spt:22 dpts:512:65535 flags:!0x16/0x02
    0     0 ACCEPT     tcp      *      *       ::/0
```

```
¬ 2001:0db8:100::2/128tcp spt:22 dpts:512:65535 flags:!0x16/0x02
    0     0 ACCEPT     icmpv6    *      *       ::/0                ::/0
    0     0 ACCEPT     tcp       *      *       ::/0                ::/0
¬        tcp spts:1024:65535 dpts:1:65535
    0     0 ACCEPT     udp       *      *       ::/0                ::/0
¬        udp spts:1024:65535 dpts:1:65535
    0     0 LOG        all       *      *       ::/0                ::/0
¬        LOG flags 0 level 7 prefix `extOUT-default:'
    0     0 DROP       all       *      *       ::/0                ::/0

Chain int2ext (1 references)
 pkts bytes target     prot opt in     out     source               destination
¬
    0     0 ACCEPT     icmpv6    *      *       ::/0                ::/0
    0     0 ACCEPT     tcp       *      *       ::/0                ::/0
¬        tcp spts:1024:65535 dpts:1:65535
    0     0 LOG        all       *      *       ::/0                ::/0
¬        LOG flags 0 level 7 prefix `int2ext:'
    0     0 DROP       all       *      *       ::/0                ::/0
    0     0 LOG        all       *      *       ::/0                ::/0
¬        LOG flags 0 level 7 prefix `int2ext-default:'
    0     0 DROP       tcp       *      *       ::/0                ::/0
    0     0 DROP       udp       *      *       ::/0                ::/0
    0     0 DROP       all       *      *       ::/0                ::/0

Chain intIN (1 references)
 pkts bytes target     prot opt in     out     source               destination
¬
    0     0 ACCEPT     all       *      *       ::/0
¬ fe80::/ffc0::
    4   384 ACCEPT     all       *      *       ::/0                ff02::/16

Chain intOUT (1 references)
 pkts bytes target     prot opt in     out     source               destination
¬
    0     0 ACCEPT     all       *      *       ::/0
¬ fe80::/ffc0::
    4   384 ACCEPT     all       *      *       ::/0                ff02::/16
    0     0 LOG        all       *      *       ::/0                ::/0
¬        LOG flags 0 level 7 prefix `intOUT-default:'
    0     0 DROP       all       *      *       ::/0                ::/0
```

Kapitel 19. Sicherheit

19.1. Sicherheit des Knoten

Es wird sehr empfohlen alle verfügbaren Patches einzuspielen sowie alle nicht benötigten Dienste zu deaktivieren. Ebenfalls sollten Sie lokales firewalling aktivieren und binden Sie die Dienste ausschließlich an benötigte IPv4/IPv6 Adressen.

Mehr Infos hierzu in späteren Versionen.

19.2. Zugangsbeschränkungen

Viele Dienste setzen die tcp_wrapper Bibliothek für die Zugangskontrolle ein. Eine Beschreibung finden Sie unter use of tcp_wrapper.

Mehr Infos hierzu in späteren Versionen.

19.3. IPv6 Sicherheitsüberwachung

Aktuell gibt es keine komfortablen Sicherheitstools mit denen man ein System über ein Netzwerk nach IPv6 relevanten Sicherheitslücken hin überprüfen kann. Weder Nessus noch irgendein kommerzieller Security Scanner ist zur Zeit dazu in der Lage, IPv6-Adressen scannen zu können.

19.3.1. Rechtsfragen

ACHTUNG: Bitte stellen Sie immer sicher, dass Sie ausschließlich ihr eigenes Netzwerk scannen oder einen Scan nur nach Erhalt einer schriftlichen Erlaubnis durchführen. Andernfalls haben sie mit rechtlichen Konsequenzen zu rechnen! ÜBERPRÜFEN Sie die Ziel-IPv6-Adresse ZWEIMAL, bevor Sie einen Scan starten.

19.3.2. Sicherheitsüberwachung mit IPv6 fähigen netcat

Mit dem IPv6 fähigen netcat (siehe IPv6+Linux-status-apps/security-auditing für Details) können Sie einen Portscan durchführen. Es wird ein Script abgearbeitet, wobei u.a. ein Port-Bereich überprüft und Banners mitprotokolliert werden. Anwendungsbeispiel:

```
# nc6 ::1 daytime
13 JUL 2002 11:22:22 CEST
```

19.3.3. Sicherheitsüberwachung mit IPv6 fähigen NMap

NMap, einer der weltweit besten Portscanner, unterstützt IPv6 seit der Version 3.10ALPHA1. Anwendungsbeispiel:

```
# nmap -6 -sT ::1
Starting nmap V. 3.10ALPHA3 ( www.insecure.org/nmap/ )
Interesting ports on localhost6 (::1):
(The 1600 ports scanned but not shown below are in state: closed)
Port       State       Service
```

```
22/tcp     open        ssh
53/tcp     open        domain
515/tcp    open        printer
2401/tcp   open        cvspserver
Nmap run completed -- 1 IP address (1 host up) scanned in 0.525 seconds
```

19.3.4. Sicherheitsüberwachung mit IPv6 fähigen strobe

Strobe ist (im Vergleich zu NMap) ein low budget Portscanner. Allerdings gibt es für Strobe einen IPv6 Patch (siehe IPv6+Linux-status-apps/security-auditing für Details). Anwendungsbeispiel:

```
# ./strobe ::1 strobe 1.05 (c) 1995-1999 Julian Assange <proff@iq.org>.
::1 2401 unassigned unknown
::1 22 ssh Secure Shell - RSA encrypted rsh
::1 515 printer spooler (lpd)
::1 6010 unassigned unknown
::1 53 domain Domain Name Server
```

Hinweis: strobe wird nicht wirklich weiterentwickelt, die abgebildete Versionsnummer ist zudem falsch.

19.3.5. Überwachungsergebnisse

Falls das Ergebnis einer Überwachung nicht Ihren IPv6 Sicherheitsrichtlinien entspricht, schließen Sie die Lücken mit Hilfe der IPv6-Firewall-Funktionalität, z.B. mit netfilter6 (siehe Firewalling/Netfilter6 für Details).

Hinweis: Detailliertere Informationen zum Thema IPv6 Sicherheit finden Sie unter folgenden Links:

- IETF drafts - IPv6 Operations (v6ops)
- RFC 3964 / Security Considerations for 6to4

Kapitel 20. Verschlüsselung und Authentifizierung

Zum Unterschied zu IPv4 ist die Verschlüsselung und die Authentifizierung ein zwingendes Feature bei IPv6. Diese Features werden normalerweise mit IPsec implementiert (das auch von IPv4 verwendet wird).

20.1. Nutzungsarten von Verschlüsselung und Authentifizierung

Zwei Arten von Verschlüsselung und Authentifzierung einer Verbindung sind möglich:

20.1.1. Transport-Modus

Der Transport-Modus ist ein Modus nur für Ende-zu-Ende-Verbindungen. Hier wird nur die Nutzlast (üblicherweise ICMP, TCP oder UDP) mit deren entsprechenden Headern verschlüsselt, wogegen der IP-Header nicht verschlüsselt wird (aber üblicherweise in die Authentifizierung eingebunden wird).

Bei Nutzung von AES-128 für Verschlüsselung und SHA1 für Authentifizierung reduziert dieser Modus die MTU um 42 Oktetts.

20.1.2. Tunnel-Modus

Der Tunnel-Modus kann einerseits für eine Ende-zu-Ende wie auch für eine Gatewas-zu-Gateway-Verbindung genutzt werden. Hier wird das komplette IP-Paket verschlüsselt und ein neuer IP-Header vorangestellt .

Dieser Modus reduziert die MTU um weitere 40 Oktetts (bei IPv6), ausgehend von der MTU des Transport-Modus.

20.2. Unterstützung im Kernel (ESP und AH)

20.2.1. Unterstützung im vanilla Linux Kernel 2.4.x

Fehlt in vanilla 2.4. In der Vergangenheit gab es einen Grund, die Linux Kernel Quellen frei von Export/Import-Kontrollgesetzen bzgl. Verschlüsselungs-Techniken zu halten. Dies ist auch ein Grund, wieso FreeS/WAN project nicht in die vanilla Quellen miteingebunden wurde.

20.2.2. Unterstützung im vanilla Linux kernel 2.6.x

Aktuelle Versionen (zum Zeitpunkt des Schreibens 2.6.9 und neuer) unterstützt IPsec für IPv4 und IPv6.

Die Implementierung wurde u.a. vom USAGI project unterstützt.

20.3. Automatischer Schlüssel-Austausch (IKE)

IPsec benötigt einen Schlüsselaustausch mit einem "Geheimnis". Dieser Vorgang wird meistens automatisch durch sogenannte IKE-Daemons durchgeführt. Diese führen ebenso die Authentifizierung der Partner durch, entweder durch ein gemeinsam bekanntes Geheimnis (auch "pre-shared secret" genannt) oder bei RSA-Schlüssel (z.B. aus X.509 Zertifikaten).

Momentan stehen (für Linux) zwei verschiedene IKE-Daemons zur Verfügung, die aber sich ziemlich in Konfiguration und Benutzung unterscheiden.

Ich präferiere "pluto" von der *S/WAN Implementierung, wei dieser eine überschaubare (und nur eine) Konfiguration.

20.3.1. IKE-Daemon "racoon"

Der IKE-Daemon "racoon" ist vom KAME-Project und auf Linux portiert worden. Aktuelle Linux-Distributionen beinhalten diesen Daemon im Paket "ipsec-tools". Zwei Programme sind für ein funktionierendes IPsec-Setup notwendig. Siehe dazu auch das Linux Advanced Routing & Traffic Control HOWTO / IPSEC.

20.3.1.1. Manipulation der IPsec SA/SP Datenbank mit dem Werkzeug "setkey"

"setkey" ist für die Definition der Security Policy (SP) im Kernel wichtig.

Datei: /etc/racoon/setkey.sh

- Beispiel für eine Ende-zu-Ende verschlüsselte Verbindung im Transport-Modus

```
#!/sbin/setkey -f
flush;
spdflush;
spdadd 2001:db8:1:1::1 2001:db8:2:2::2 any -P out ipsec esp/transport//require;
spdadd 2001:db8:2:2::2 2001:db8:1:1::1 any -P in  ipsec esp/transport//require;
```

- Beispiel für eine Ende-zu-Ende verschlüsselte Verbindung im Tunnel-Modus

```
#!/sbin/setkey -f
flush;
spdflush;
spdadd 2001:db8:1:1::1 2001:db8:2:2::2 any -P out ipsec
¬ esp/tunnel/2001:db8:1:1::1-2001:db8:2:2::2/require;
spdadd 2001:db8:2:2::2 2001:db8:1:1::1 any -P in  ipsec
¬ esp/tunnel/2001:db8:2:2::2-2001:db8:1:1::1/require;
```

Beim anderen Partner ist "in" mit "out" zu vertauschen.

20.3.1.2. Konfiguration des IKE-Daemon "racoon"

"racoon" benötigt eine Konfigurationsdatei zur Ausführung. Es beinhaltet zu der Security Policy entprechenden Einstellungen, welche vorher mit "setkey" definiert wurde.

Datei: /etc/racoon/racoon.conf

```
# Racoon IKE daemon configuration file.
# See 'man racoon.conf' for a description of the format and entries.
path include "/etc/racoon";
path pre_shared_key "/etc/racoon/psk.txt";

listen
{
        isakmp 2001:db8:1:1::1;
}
```

```
remote 2001:db8:2:2::2
{
        exchange_mode main;
        lifetime time 24 hour;
        proposal
        {
                encryption_algorithm 3des;
                hash_algorithm md5;
                authentication_method pre_shared_key;
                dh_group 2;
        }
}

# gateway-to-gateway
sainfo address 2001:db8:1:1::1 any address 2001:db8:2:2::2 any
{
        lifetime time 1 hour;
        encryption_algorithm 3des;
        authentication_algorithm hmac_md5;
        compression_algorithm deflate;
}

sainfo address 2001:db8:2:2::2 any address 2001:db8:1:1::1 any
{
        lifetime time 1 hour;
        encryption_algorithm 3des;
        authentication_algorithm hmac_md5;
        compression_algorithm deflate;
}
```

Zudem muss das gemeinsame Geheimnis definiert werden:

Datei: /etc/racoon/psk.txt

```
# file for pre-shared keys used for IKE authentication
# format is: 'identifier' 'key'

2001:db8:2:2::2 verysecret
```

20.3.1.3. IPsec mit IKE-Daemon "racoon" starten

Zum Schluss muss der Daemon gestartet werden. Beim ersten Mal sollte Debug- & Vordergrund-Modus aktiviert werden. Das folgende Beispiel zeigt eine erfolgreiche Aushandlung von IKE-Phase 1 (ISAKMP-SA) und 2 (IPsec-SA):

```
# racoon -F -v -f /etc/racoon/racoon.conf
Foreground mode.
2005-01-01 20:30:15: INFO: @(#)ipsec-tools 0.3.3 (http://ipsec-tools.sourceforge.net)
2005-01-01 20:30:15: INFO: @(#)This product linked
¬ OpenSSL 0.9.7a Feb 19 2003 (http://www.openssl.org/)
2005-01-01 20:30:15: INFO: 2001:db8:1:1::1[500] used as isakmp port (fd=7)
2005-01-01 20:31:06: INFO: IPsec-SA request for 2001:db8:2:2::2
¬ queued due to no phase1 found.
2005-01-01 20:31:06: INFO: initiate new phase 1 negotiation:
¬ 2001:db8:1:1::1[500]<=>2001:db8:2:2::2[500]
2005-01-01 20:31:06: INFO: begin Identity Protection mode.
2005-01-01 20:31:09: INFO: ISAKMP-SA established
¬ 2001:db8:1:1::1[500]-2001:db8:2:2::2[500] spi:da3d3693289c9698:ac039a402b2db401
2005-01-01 20:31:09: INFO: initiate new phase 2 negotiation:
```

```
¬ 2001:6f8:900:94::2[0]<=>2001:db8:2:2::2[0]
2005-01-01 20:31:10: INFO: IPsec-SA established:
¬ ESP/Tunnel 2001:db8:2:2::2->2001:db8:1:1::1 spi=253935531(0xf22bfab)
2005-01-01 20:31:10: INFO: IPsec-SA established:
¬ ESP/Tunnel 2001:db8:1:1::1->2001:db8:2:2::2 spi=175002564(0xa6e53c4)
```

Jede Richtung bekommt einen eigenen SPI (wie im IPsec-Standard definiert). Mit "tcpdump" kann an der entprechenden Schnittstelle dann das Ergebnis eines IPv6-pings gesehen werden:

```
20:35:55.305707 2001:db8:1:1::1 > 2001:db8:2:2::2: ESP(spi=0x0a6e53c4,seq=0x3)
20:35:55.537522 2001:db8:2:2::2 > 2001:db8:1:1::1: ESP(spi=0x0f22bfab,seq=0x3)
```

Wie erwartet, werden die ausgehandelten SPIs angezeigt.

Mit "setkey" werden die aktiven Parameter angezeigt:

```
# setkey -D
2001:db8:1:1::1 2001:db8:2:2::2
        esp mode=tunnel spi=175002564(0x0a6e53c4) reqid=0(0x00000000)
        E: 3des-cbc  bd26bc45 aea0d249 ef9c6b89 7056080f 5d9fa49c 924e2edd
        A: hmac-md5  60c2c505 517dd8b7 c9609128 a5efc2db
        seq=0x00000000 replay=4 flags=0x00000000 state=mature
        created: Jan  1 20:31:10 2005   current: Jan  1 20:40:47 2005
        diff: 577(s)    hard: 3600(s)   soft: 2880(s)
        last: Jan  1 20:35:05 2005      hard: 0(s)      soft: 0(s)
        current: 540(bytes)     hard: 0(bytes)  soft: 0(bytes)
        allocated: 3    hard: 0 soft: 0
        sadb_seq=1 pid=22358 refcnt=0
2001:db8:2:2::2 2001:db8:1:1::1
        esp mode=tunnel spi=253935531(0x0f22bfab) reqid=0(0x00000000)
        E: 3des-cbc  c1ddba65 83debd62 3f6683c1 20e747ac 933d203f 4777a7ce
        A: hmac-md5  3f957db9 9adddc8c 44e5739d 3f53ca0e
        seq=0x00000000 replay=4 flags=0x00000000 state=mature
        created: Jan  1 20:31:10 2005   current: Jan  1 20:40:47 2005
        diff: 577(s)    hard: 3600(s)   soft: 2880(s)
        last: Jan  1 20:35:05 2005      hard: 0(s)      soft: 0(s)
        current: 312(bytes)     hard: 0(bytes)  soft: 0(bytes)
        allocated: 3    hard: 0 soft: 0
        sadb_seq=0 pid=22358 refcnt=0
```

20.3.2. IKE-Daemon "pluto"

Der IKE-Daemon "pluto" ist in den Paketen der *S/WAN-Projekte beinhaltet. Das *S/WAN-Projekt startete zu Anfags als FreeS/WAN. Leider wurde die Weiterentwicklung von FreeS/WAN in 2004 eingestellt. Aufgrund der langsamen Entwicklungsgeschwindigkeit in der Vergangenheit entstanden zwei Spin-Offs: strongSwan und Openswan. Heutzutage stehen installationsfertige Pakete bereit, u.a. von Openswan (in Fedora Core 3 beinhaltet).

Ein großer Unterschied zu "racoon" ist, dass nur eine Konfigurationsdatei notwendig ist. Zudem steht ein initscript für automatisches Starten beim Booten zur Verfügung.

20.3.2.1. Konfiguration des IKE-Daemon "pluto"

Die Konfiguration ist der zu IPv4 sehr ähnlich, nur eine wichtige Option ist notwendig.

Datei: /etc/ipsec.conf

```
# /etc/ipsec.conf - Openswan IPsec configuration file
#
# Manual:     ipsec.conf.5
version 2.0     # conforms to second version of ipsec.conf specification

# basic configuration
config setup
        # Debug-logging controls:  "none" for (almost) none, "all" for lots.
        # klipsdebug=none
        # plutodebug="control parsing"

#Disable Opportunistic Encryption
include /etc/ipsec.d/examples/no_oe.conf

conn ipv6-p1-p2
        connaddrfamily=ipv6       # Important for IPv6!
        left=2001:db8:1:1::1
        right=2001:db8:2:2::2
        authby=secret
        esp=aes128-sha1
        ike=aes128-sha-modp1024
        type=transport
        #type=tunnel
        compress=no
        #compress=yes
        auto=add
        #auto=start
```

Zudem muss das gemeinsame Geheimnis definiert werden:

Datei: /etc/ipsec.secrets

```
2001:db8:1:1::1 2001:db8:2:2::2 : PSK      "verysecret"
```

20.3.2.2. IPsec mit IKE daemon "pluto" starten

Wenn die Installation von Openswan erfolgreich war, sollte ein initscript zum Starten von IPsec zur Verfügung stehen. Dann einfach auf jedem Partner folgendes ausführen:

```
# /etc/rc.d/init.d/ipsec start
```

Danach kann die Verbindung auf einem Partner gestartet werden. Wenn im folgenden die Zeile "IPsec SA established" erscheint, hat die Aushandlung funktioniert.

```
# ipsec auto --up ipv6-peer1-peer2
104 "ipv6-p1-p2" #1: STATE_MAIN_I1: initiate
106 "ipv6-p1-p2" #1: STATE_MAIN_I2: sent MI2, expecting MR2
108 "ipv6-p1-p2" #1: STATE_MAIN_I3: sent MI3, expecting MR3
004 "ipv6-p1-p2" #1: STATE_MAIN_I4: ISAKMP SA established
112 "ipv6-p1-p2" #2: STATE_QUICK_I1: initiate
004 "ipv6-p1-p2" #2: STATE_QUICK_I2: sent QI2,
¬ IPsec SA established {ESP=>0xa98b7710 <0xa51e1f22}
```

Weil *S/WAN und setkey/racoon die gleiche IPsec-Implementation im Linux kernel 2.6.x benutzen, zeigt "setkey" auch hier die aktiven Parameter:

```
# setkey -D
2001:db8:1:1::1 2001:db8:2:2::2
        esp mode=transport spi=2844489488(0xa98b7710) reqid=16385(0x00004001)
        E: aes-cbc  082ee274 2744bae5 7451da37 1162b483
```

```
        A: hmac-sha1  b7803753 757417da 477b1c1a 64070455 ab79082c
        seq=0x00000000 replay=64 flags=0x00000000 state=mature
        created: Jan  1 21:16:32 2005   current: Jan  1 21:22:20 2005
        diff: 348(s)    hard: 0(s)      soft: 0(s)
        last:                           hard: 0(s)      soft: 0(s)
        current: 0(bytes)       hard: 0(bytes)  soft: 0(bytes)
        allocated: 0    hard: 0 soft: 0
        sadb_seq=1 pid=23825 refcnt=0
2001:db8:2:2::2 2001:db8:1:1::1
        esp mode=transport spi=2770214690(0xa51e1f22) reqid=16385(0x00004001)
        E: aes-cbc  6f59cc30 8d856056 65e07b76 552cac18
        A: hmac-sha1  c7c7d82b abfca8b1 5440021f e0c3b335 975b508b
        seq=0x00000000 replay=64 flags=0x00000000 state=mature
        created: Jan  1 21:16:31 2005   current: Jan  1 21:22:20 2005
        diff: 349(s)    hard: 0(s)      soft: 0(s)
        last:                           hard: 0(s)      soft: 0(s)
        current: 0(bytes)       hard: 0(bytes)  soft: 0(bytes)
        allocated: 0    hard: 0 soft: 0
        sadb_seq=0 pid=23825 refcnt=0
```

20.4. Anmerkungen:

Bei Linux Kernel 2.6.x kann der IPsec-Status und die Policy auch mit "ip" angezeigt werden:

```
# ip xfrm policy
...

# ip xfrm state
...
```

Kapitel 21. Quality of Service (QoS)

IPv6 unterstützt QoS durch die Anwendung von Flow Labels und Traffic Classes. QoS kann mittels "tc" (im Paket "iproute" enthalten) kontrolliert werden.

Zusätzliche Infos:

- RFC 3697 / IPv6 Flow Label Specification

Mehr Infos hierzu in späteren Versionen.

Kapitel 22. Hinweise zu IPv6 kompatiblen Daemons

Im folgenden Kapitel werden einige Hinweise zu IPv6 kompatiblen Daemons gegeben.

22.1. Berkeley Internet Name Domain (BIND) daemon "named"

Seit der Version 9 wird IPv6 unterstützt. Setzen Sie immer die neuest verfügbare Version ein. Zumindest muss Version 9.1.3 eingesetzt werden, da ältere Versionen Sicherheitslöcher beinhalten können, die von Remote entsprechend ausgenutzt werden können.

22.1.1. Auf IPv6 Adressen hören

Anmerkung: Im Gegensatz zu IPv4 können bei aktuellen Versionen Server Sockets nicht an dedizierte IPv6 Adressen gebunden werden, es ist folglich *jede* oder *keine* Adresse gültig. Da dies ein Sicherheitsproblem sein kann, lesen Sie diesbezüglich ebenfalls den Abschnitt Access Control Lists (ACL) weiter unten!

22.1.1.1. BIND named konfigurieren, damit er auf IPv6 Adressen antwortet

Folgende Optionen müssen geändert werden, damit IPv6 aktiviert wird

```
options {
        # sure other options here, too
        listen-on-v6 { any; };
};
```

Nach einem Neustart (des Dienstes) sollte z.B. Folgendes zu sehen sein:

```
# netstat -lnptu |grep "named\W*$"
tcp 0 0 :::53         :::*      LISTEN 1234/named
¬ # incoming TCP requests
udp 0 0 1.2.3.4:53    0.0.0.0:*        1234/named
¬ # incoming UDP requests to IPv4 1.2.3.4
udp 0 0 127.0.0.1:53  0.0.0.0:*        1234/named
¬ # incoming UDP requests to IPv4 localhost
udp 0 0 0.0.0.0:32868 0.0.0.0:*        1234/named
¬ # dynamic chosen port for outgoing queries
udp 0 0 :::53         :::*             1234/named
¬ # incoming UDP request to any IPv6
```

Ein kleiner Test sieht wie folgt aus:

```
# dig localhost @::1
```

und sollte Ihnen ein Ergebnis anzeigen...

22.1.1.2. BIND named konfigurieren, damit er auf IPv6 Adressen nicht antwortet

Folgende Optionen müssen geändert werden, damit IPv6 deaktiviert wird:

```
options {
        # sure other options here, too
        listen-on-v6 { none; };
};
```

22.1.2. Access Control Lists (ACL) mit IPv6 Unterstützung

ACLs mit IPv6 Adressen sind realisierbar und sollten wann immer möglich eingesetzt werden. Ein Beispiel:

```
acl internal-net {
        127.0.0.1;
        1.2.3.0/24;
        2001:0db8:100::/56;
        ::1/128;
        ::ffff:1.2.3.4/128;
};
acl ns-internal-net {
        1.2.3.4;
        1.2.3.5;
        2001:0db8:100::4/128;
        2001:0db8:100::5/128;
};
```

Diese ACLs können für Client-Anfragen und Zonentransfers zu Secondary Nameserver eingesetzt werden. Es kann auch unterbunden werden, dass ihr Caching-Nameserver mittels IPv6 von der Außenwelt verwendet wird.

```
options {
        # sure other options here, too
        listen-on-v6 { none; };
        allow-query { internal-net; };
        allow-transfer { ns-internal-net; };
};
```

Es ist ebenfalls möglich, dass die Optionen *allow-query* und *allow-transfer* bei den meisten Single-Zonen-Definitionen verwendet werden.

22.1.3. Anfragen mit festen IPv6 Adressen senden

Diese Option ist nicht verpflichtend, ev. aber benötigt:

```
query-source-v6 address <ipv6address|*> port <port|*>;
```

22.1.4. Pro Zone definierte feste IPv6 Adressen

Es ist möglich pro Zone mehrere IPv6 Adressen zu definieren.

22.1.4.1. Transfer source Adresse

Die Transfer source Adresse wird für ausgehende Zonentransfers verwendet:

```
transfer-source-v6 <ipv6addr|*> [port port];
```

22.1.4.2. Notify source Adresse

Die Notify source Adresse wird für ausgehende notify Mitteilungen verwendet:

```
notify-source-v6 <ipv6addr|*> [port port];
```

22.1.5. IPv6 DNS zone files Beispiele

Einige Informationen finden Sie auch unter IPv6 DNS Setup Information (article). Eventuell ebenfalls hilfreich ist folgendes Tool: IPv6 Reverse DNS zone builder for BIND 8/9 (webtool).

22.1.6. IPv6 bezogene DNS-Daten bereitstellen

Für IPv6 wurden neue Reverse Lookup Arten und Root Zonen definiert:

- AAAA und reverse IP6.INT: beschrieben in RFC 1886 / DNS Extensions to support IP version 6 sowie seit BIND Version 4.9.6 in Verwendung
- A6, DNAME (WURDE ABGELEHNT!) und reverse IP6.ARPA: beschrieben in RFC 2874 / DNS Extensions to Support IPv6 Address Aggregation and Renumbering sowie seit BIND 9 in Verwendung. Informationen zum aktuellen Stand sind unter Domain Name System Extension (dnsext) zu finden.

Mehr Inhalt zu diesem Thema wird eventuell in späteren Versionen eingearbeitet, inzwischen können Sie in den RFCs und in folgenden Quellen nachlesen:

- AAAA und reverse IP6.INT: IPv6 DNS Setup Information
- A6, DNAME (WURDE ABGELEHNT!) und reverse IP6.ARPA: lesen Sie im Kapitel 4 und 6 des BIND 9 Administrator Reference Manual (ARM) nach, welches mit dem bind-Paket mitgeliefert wird. Sie können es auch hier lesen: BIND manual version 9.3

Da IP6.INT (ebenfalls) ABGELEHNT WURDE, (jedoch nach wie vor in Verwendung ist,) muss ein DNS Server, der IPv6 Informationen anbieten will, beide reverse Zonen bereitstellen.

22.1.6.1. Aktuell beste Praxis

Da es mit den neuen Formaten noch Probleme gibt, ist die aktuell beste Praxis:

Vorwärts-Auflösung mit:

- AAAA

Rückwärts-Auflösung mit:

- Reverse nibble format für die Zone ip6.int (FÜR RÜCKWÄRTSKOMPATIBILITÄT)
- Reverse nibble format für die Zone ip6.arpa (EMPFHOHLEN)

22.1.7. IPv6 Verbindung überprüfen

Ob BIND auf einen IPv6 socket hört bzw. IPv6 Daten bereitstellt, können Sie anhand folgender Beispiele überprüfen.

22.1.7.1. IPv6 Verbindung durch ACL abgelehnt

Eine IPv6 Verbindung kann durch Angabe eines dedizierten Server, der abgefragt werden soll, erzwungen werden:

```
$ host -t aaaa www.6bone.net 2001:0db8:200:f101::1
Using domain server:
Name: 2001:0db8:200:f101::1
Address: 2001:0db8:200:f101::1#53
Aliases:

Host www.6bone.net. not found: 5(REFUSED)
```

Ein entsprechender Log-Eintrag sieht wie folgt aus:

```
Jan 3 12:43:32 gate named[12347]: client
¬ 2001:0db8:200:f101:212:34ff:fe12:3456#32770:
 query denied
```

Wenn Sie diesen Eintrag in der Logdatei finden, prüfen Sie, ob von diesem Client Anfragen akzeptiert werden sollen und ggf. ändern Sie Ihre ACL Konfiguration.

22.1.7.2. Erfolgreiche IPv6 Verbindung

Eine erfolgreiche IPv6 Verbindung sieht wie folgt aus:

```
$ host -t aaaa www.6bone.net 2001:0db8:200:f101::1
Using domain server:
Name: 2001:0db8:200:f101::1
Address: 2001:0db8:200:f101::1#53
Aliases:

www.6bone.net. is an alias for 6bone.net.
6bone.net. has AAAA address 3ffe:b00:c18:1::10
```

22.2. Internet super daemon (xinetd)

IPv6 wird ungefähr seit der xinetd Version 1.8.9 unterstützt. Verwenden sie immer die neueste Version, zumindest aber Version 2.3.3, da ältere Versionen Sicherheitslöcher beinhalten können, die von Remote entsprechend ausgenutzt werden können.

Einige Linux Distributionen beinhalten ein separates IPv6 kompatibles Paket des xinetd, bei anderen Distributionen wird der IPv6 kompatible xinetd mit folgender Variable zumeist in der Datei /etc/sysconfig/network (bei Red Hat kompatible Distributionen) gestartet: NETWORKING_IPV6="yes". In neuere Versionen unterstützt eine Binärdatei sowohl IPv4 als auch IPv6.

Wenn Sie nun einen "eingebauten" Service wie z.B. daytime durch folgende Änderung der Konfigurationsdatei /etc/xinetd.d/daytime aktivieren

```
# diff -u /etc/xinetd.d/daytime.orig /etc/xinetd.d/daytime
--- /etc/xinetd.d/daytime.orig Sun Dec 16 19:00:14 2001
+++ /etc/xinetd.d/daytime Sun Dec 16 19:00:22 2001
@@ -10,5 +10,5 @@
        protocol = tcp
        user = root
        wait = no
-       disable = yes
+       disable = no
 }
```

dann sollten Sie nach einem Neustart des xinetd-Dienstes z.B. folgendes positive Ergebnis sehen:

```
# netstat -lnptu -A inet6 |grep "xinetd*"
tcp 0 0 ::ffff:192.168.1.1:993  :::*  LISTEN  12345/xinetd-ipv6
tcp 0 0 :::13                   :::*  LISTEN  12345/xinetd-ipv6 <- service
¬ daytime/tcp
tcp 0 0 ::ffff:192.168.1.1:143  :::*  LISTEN  12345/xinetd-ipv6
```

Das Beispiel zeigt auch die xinetd Dienste IMAP und IMAP-SSL, die nur auf IPv4 Adressen hören.

Hinweis: frühere Versionen hatten ein Problem, dass der nur für IPv4 kompilierte xinetd nicht bei einem IPv6-aktivierten Knoten startete, und eine IPv6-aktivierte nicht bei einem Knoten, der nur IPv4 aktiv hatte. Dies sollte aber mindestens seit Version 2.3.11 gefixt sein.

22.3. Webserver Apache2 (httpd2)

IPv6 wird beim Apache Webserver durch die Entwickler seit der Version 2.0.14 unterstützt. Verfügbare Patches für die alte 1.3.x Serie sind inzwischen nicht mehr aktuell und sollten nicht mehr in öffentlich zugänglichen Umgebungen eingesetzt werden. Verfügbar sind die Patches noch unter KAME / Misc.

22.3.1. Auf IPv6 Adressen hören

Anmerkung: Virtuelle Hosts mit IPv6 Adressen sind bis zur Version 2.0.28 nicht operabel (es gibt für die Version 2.0.28 einen Patch). Testen Sie aber immer zuerst die neueste Version, da ältere Versionen mitunter auch Sicherheitsprobleme mit sich bringen können.

22.3.1.1. Virtueller Host mit IPv6 Adresse

```
Listen [2001:0db8:100::1]:80
<VirtualHost [2001:0db8:100::1]:80>
        ServerName ipv6only.yourdomain.yourtopleveldomain
        # ...sure more config lines
</VirtualHost>
```

22.3.1.2. Virtueller Host mit IPv4 und IPv6 Adresse

```
Listen [2001:0db8:100::2]:80
Listen 1.2.3.4:80
<VirtualHost [2001:0db8:100::2]:80 1.2.3.4:80>
        ServerName ipv6andipv4.yourdomain.yourtopleveldomain
        # ...sure more config lines
</VirtualHost>
```

Das Ergebnis sollten nach einen Neustart des Dienstes etwa Folgendes sein:

```
# netstat -lnptu |grep "httpd2\W*$"
tcp 0 0 1.2.3.4:80          0.0.0.0:* LISTEN 12345/httpd2
tcp 0 0 2001:0db8:100::1:80 :::*      LISTEN 12345/httpd2
tcp 0 0 2001:0db8:100::2:80 :::*      LISTEN 12345/httpd2
```

Für einfache Tests können Sie auf das bereits gezeigte telnet-Beispiel zurückgreifen.

22.3.1.3. Zusätzliche Anmerkungen

Apache2 unterstützt eine Methode namens "sendfile", um die Auslieferung von Datenn zu beschleunigen. Einige NIC-Treiber unterstützen auch offline das Berechnen der Checksumme. In einigen Fällen kann dies zu Verbindungsproblemen und ungültigen TCP-Checksummen führen. In diesen Fällen ist "sendfile" zu deaktivieren, entweder durch Rekompilieren unter der Benützung der configure-Option "--without-sendfile"

oder durch Benützung der Direktive "EnableSendfile off" in der Konfigurationsdatei.

22.4. Router Advertisement Daemon (radvd)

Der Router Advertisement Daemon ist auf einem LAN dann sehr sinnvoll, wenn die Clients automatisch konfiguriert werden sollen. Der Daemon selbst sollte auf einem Linux Gateway Router eingerichtet sein (es hat nicht notwendigerweise das default IPv4 Gateway zu sein, Vorsicht also wer am LAN Router Advertisements versendet).

Sie können einige Flags und Informationen im Advertisement spezifizieren. Allgemein werden verwendet:

- Präfix (notwendige Angabe)
- Lebensdauer des Präfix
- Intervall der Advertisements (optional)

Nach der korrekten Konfiguration sendet der Daemon die Advertisements über angegebene Interfaces. Die Clients empfangen die Advertisements und konfigurieren automatisch Ihre Adressen mit dem empfangenen Präfix und der Default-Route.

22.4.1. radvd konfigurieren

22.4.1.1. Einfache Konfiguration

Die Konfigurationsdatei des radvd ist normalerweise die Datei /etc/radvd.conf. Eine einfache Konfiguration sieht wie folgt aus:

```
interface eth0 {
        AdvSendAdvert on;
        MinRtrAdvInterval 3;
        MaxRtrAdvInterval 10;
        prefix 2001:0db8:0100:f101::/64 {
                AdvOnLink on;
                AdvAutonomous on;
                AdvRouterAddr on;
        };
};
```

Als Ergebnis auf der Client-Seite ergibt sich hieraus:

```
# ip -6 addr show eth0
3: eth0: <BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast qlen 100
    inet6 2001:0db8:100:f101:2e0:12ff:fe34:1234/64 scope global dynamic
       valid_lft 2591992sec preferred_lft 604792sec
    inet6 fe80::2e0:12ff:fe34:1234/10 scope link
```

Ein hoher Wert für die Lebensdauer wurde verwendet, da der Wert nicht manuell konfiguriert wurde.

22.4.1.2. Spezielle 6to4 Konfiguration

Seit der Version 0.6.2pl3 wird die automatische (Neu)-Erstellung des Präfixes abhängig von der IPv4 Adresse eines angegebenen Interfaces unterstützt. Dies kann dazu eingesetzt werden, die Advertisements dann in einem LAN zu verteilen, nachdem das 6to4 tunneling geändert wurde. Zumeist eingesetzt wird dies hinter einem dynamischen dial-on-demand Linux Router. Wegen der sicherlich kürzeren Lebensdauer dieser Präfixe (nach jedem dial-up ist ein anderes Präfix gültig), wird der Wert der Lebensdauer auf einen minimalen Wert gesetzt:

```
interface eth0 {
        AdvSendAdvert on;
        MinRtrAdvInterval 3;
        MaxRtrAdvInterval 10;
        prefix 0:0:0:f101::/64 {
                AdvOnLink off;
                AdvAutonomous on;
                AdvRouterAddr on;
                Base6to4Interface ppp0;
                AdvPreferredLifetime 20;
                AdvValidLifetime 30;
        };
};
```

Das Ergebnis auf Clientseite ist (unter der Annahme, dass ppp0 die lokale IPv4 Adresse 1.2.3.4 hat):

```
# /sbin/ip -6 addr show eth0
3: eth0: <BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast qlen 100
   inet6 2002:0102:0304:f101:2e0:12ff:fe34:1234/64 scope global dynamic
      valid_lft 22sec preferred_lft 12sec
   inet6 fe80::2e0:12ff:fe34:1234/10 scope link
```

Da eine kurze Lebensdauer definiert wurde, wird das Präfix bald verworfen werden, sollte kein entsprechendes Advertisement empfangen werden.

Achtung: wenn keine spezielle 6to4-Unterstützung der initscripts benutzt wird, ist eine spezielle Route am internen Interface des Routers notwendig, sonst gibt es Probleme bei eingehenden Paketen. Für das gezeigte Beispiel lautet diese:

```
# /sbin/ip -6 route add 2002:0102:0304:f101::/64 dev eth0 metric 1
```

Diese Route muß jedesmal, wenn der Prefix wechselt, ersetzt werden. Die ist dann der Fall, wenn das Dial-Up-Interface eine neue IPv4-Adresse bekommen hat.

22.4.2. Fehlersuche

Mit dem Programm "radvdump" können Sie gesendete und empfangene Advertisements detailliert betrachten. Die Anwendung ist einfach:

```
# radvdump
Router advertisement from fe80::280:c8ff:feb9:cef9 (hoplimit 255)
        AdvCurHopLimit: 64
        AdvManagedFlag: off
        AdvOtherConfigFlag: off
        AdvHomeAgentFlag: off
        AdvReachableTime: 0
        AdvRetransTimer: 0
        Prefix 2002:0102:0304:f101::/64
                AdvValidLifetime: 30
                AdvPreferredLifetime: 20
                AdvOnLink: off
                AdvAutonomous: on
                AdvRouterAddr: on
        Prefix 2001:0db8:100:f101::/64
                AdvValidLifetime: 2592000
                AdvPreferredLifetime: 604800
                AdvOnLink: on
                AdvAutonomous: on
                AdvRouterAddr: on
        AdvSourceLLAddress: 00 80 12 34 56 78
```

Im Output wird jedes Advertisement in einem lesbarem Format dargestellt. Zu sehen sollten die von Ihnen eingestellten Werte sein; falls dem nicht so ist, wurde das Advertisement eventuell nicht von Ihrem radvd gesendet... (für die Rückverfolgung des Routers können Sie die LLAddress, die MAC Adresse des Routers, verwenden...)

22.5. Dynamic Host Configuration v6 Server (dhcp6s)

DHCPv6 kann für stateful Konfiguration benutzt werden. Der Daemon selbst muß nicht unbedingt auf dem Linux-Standard-Router laufen.

Man kann hier mehr Informationen als bei radvd spezifizieren. Die meisten sind denen des IPv4 DHCP-Servers ähnlich.

Nach einer passenden Konfiguration reagiert der Daemon aur empfangene IPv6-Multicast-Pakete, die von einem Client an die Adresse ff02::16 gesendet werden.

22.5.1. Konfiguration des DHCPv6-Servers (dhcp6s)

22.5.1.1. Einfache Konfiguration

Die Konfigurationsdatei des dhcp6s ist normalerweise /etc/dhcp6s.conf. Ein einfaches Beispiel sieht wie folgt aus:

```
interface eth0 {
        server-preference 255;
        renew-time 60;
        rebind-time 90;
        prefer-life-time 130;
        valid-life-time 200;
        allow rapid-commit;
        option dns_servers 2001:db8:0:f101::1 sub.domain.example;
        link AAA {
                range 2001:db8:0:f101::1000 to 2001:db8:0:f101::ffff/64;
                prefix 2001:db8:0:f101::/64;
        };
};
```

22.5.2. Konfiguration des DHCPv6-Client (dhcp6s)

22.5.2.1. Einfache Konfiguration

Die Konfigurationsdatei von dhcp6c ist normalerweise /etc/dhcp6c.conf. Ein einfaches Beispiel sieht wie folgt aus:

```
interface eth0 {
        send rapid-commit;
        request domain-name-servers;
};
```

22.5.3. Benutzung

22.5.3.1. dhcp6s

Starten des Servers, z.B. durch

```
# service dhcp6s start
```

22.5.3.2. dhcp6c

Starten des Clients im Vordergrund, z.B. durch

```
# # dhcp6c -f eth0
...
```

22.5.4. Fehlersuche

22.5.4.1. dhcp6s

Der Server hat einen Vordergrund und zwei Debug-Schalter (von denen beide benutzt werden sollten), hier ein Beispiel:

```
# dhcp6c -d -D -f eth0
```

22.5.4.2. dhcp6c

Der Client hat einen Vordergrund und zwei Debug-Schalter, hier ein Beispiel:

```
# dhcp6c -d -f eth0
Oct/03/2005 17:18:16 dhcpv6 doesn't support hardware type 776
Oct/03/2005 17:18:16 doesn't support sit0 address family 0
Oct/03/2005 17:18:16 netlink_recv_rtgenmsg error
Oct/03/2005 17:18:16 netlink_recv_rtgenmsg error
Oct/03/2005 17:18:17 status code for this address is: success
Oct/03/2005 17:18:17 status code: success
Oct/03/2005 17:18:17 netlink_recv_rtgenmsg error
Oct/03/2005 17:18:17 netlink_recv_rtgenmsg error
Oct/03/2005 17:18:17 assigned address 2001:db8:0:f101::1002 prefix len is not in any RAs prefix l
Oct/03/2005 17:18:17 renew time 60, rebind time 9
```

Bemerkung: die netlink-Fehlermeldungen haben keinen Einfluß auf die Funktionalität.

22.6. tcp_wrapper

Mit der tcp_wrapper Programmbibliothek können Sie Ihre Dienste gegen Missbrauch schützen.

22.6.1. Filter-Funktionalität

Sie können tcp_wrapper für folgende Zwecke einsetzen:

- Nach Source-Adressen filtern (IPv4 oder IPv6)
- Nach Benutzern filtern (benötigt einen aktiven ident Daemon auf der Client-Seite)

22.6.2. Welches Programm benützt tcp_wrapper

Folgende Programme sind bekannt:

- Jeder Dienst, der durch den xinetd aufgerufen wird (und wenn der xinetd mit der tcp_wrapper Bibliothek kompiliert wurde)
- sshd (wenn der mit der tcp_wrapper Bibliothek kompiliert wurde)

22.6.3. Anwendung

Der tcp_wrapper wird durch zwei Dateien konfiguriert und kontrolliert: /etc/hosts.allow sowie /etc/hosts.deny. Weitere Informationen finden Sie mit:

```
$ man hosts.allow
```

22.6.3.1. Beispiel für /etc/hosts.allow

In dieser Datei wird ein Dienst pro Zeile eingetragen, der positiv gefiltert werden soll (d.h. Verbindungen werden erlaubt).

```
sshd:           1.2.3. [2001:0db8:100:200::]/64
daytime-stream: 1.2.3. [2001:0db8:100:200::]/64
```

Achtung: es existieren fehlerhafte Implementierungen, welche folgende fehlerhafte IPv6-Netzwerk-Beschreibung unterstützen: [2001:0db8:100:200::/64]. Hoffentlich werden diese Versionen bald gefixt.

22.6.3.2. Beispiel für /etc/hosts.deny

In dieser Datei werden alle Einträge negativ gefiltert. Und normalerweise sollen alle Verbindungen unterbunden werden:

```
ALL: ALL
```

Sie können bei Bedarf obige Standardzeile auch durch Folgende ersetzen, jedoch wird dadurch bei zu vielen Verbindungen in kurzer Zeitz eine DoS Angriff möglich (Last des Mailers sowie des Spool-Verzeichnisses). Ein logwatch ist somit wahrscheinlich die bessere Lösung.

```
ALL: ALL: spawn (echo "Attempt from %h %a to %d at `date`"
 | tee -a /var/log/tcp.deny.log | mail root@localhost)
```

22.6.4. Protokollierung

Entsprechend der Syslog Daemon Konfiguration in der Datei /etc/syslog.conf protokolliert der tcp_wrapper normalerweise in die Datei /var/log/secure.

22.6.4.1. Abgelehnte Verbindung

Das Logging einer abgelehnten IPv4-Verbindung zu einem durch den xinetd überwachten Daytime Dienst sieht wie folgt aus:

```
Jan 2 20:40:44 gate xinetd-ipv6[12346]: FAIL: daytime-stream libwrap
¬ from=::ffff:1.2.3.4
```

```
Jan 2 20:32:06 gate xinetd-ipv6[12346]: FAIL: daytime-stream libwrap
 from=2001:0db8:100:200::212:34ff:fe12:3456
```

Das Logging einer abgelehnten IPv4-Verbindung zu einem durch den xinetd überwachten sshd Daemon (auf IPv4 und IPv6 auf Verbindungen wartend) sieht wie folgt aus:

```
Jan 2 20:24:17 gate sshd[12345]: refused connect from ::ffff:1.2.3.4
¬ (::ffff:1.2.3.4)
Jan 2 20:39:33 gate sshd[12345]: refused connect
 from 2001:0db8:100:200::212:34ff:fe12:3456
¬ (2001:0db8:100:200::212:34ff:fe12:3456)
```

22.6.4.2. Akzeptierte Verbindung

Das Logging einer akzeptierten IPv4-Verbindung zu einem durch den xinetd überwachten Daytime Dienst sieht wie folgt aus:

```
Jan 2 20:37:50 gate xinetd-ipv6[12346]: START: daytime-stream pid=0
¬ from=::ffff:1.2.3.4
Jan 2 20:37:56 gate xinetd-ipv6[12346]: START: daytime-stream pid=0
 from=2001:0db8:100:200::212:34ff:fe12:3456
```

Das Logging einer akzeptierten IPv4-Verbindung zu einem auf zwei Ports hörenden sshd sieht wie folgt aus:

```
Jan 2 20:43:10 gate sshd[21975]: Accepted password for user from ::ffff:1.2.3.4
¬ port 33381 ssh2
Jan 2 20:42:19 gate sshd[12345]: Accepted password for user
 from 2001:0db8:100:200::212:34ff:fe12:3456 port 33380 ssh2
```

22.7. vsftpd

22.7.1. Auf IPv6-Adressen lauschen

Editiere die Konfigurationsdatei, üblicherweise /etc/vsftpd/vsftpd.conf, und setze die Option für das "listen" wie folgt:

```
listen_ipv6=yes
```

Mehr ist nicht zu tun.

22.8. proftpd

22.8.1. Auf IPv6-Adressen lauschen

Editiere die Konfigurationsdatei, üblicherweise /etc/proftpd.conf, allerdings ist hier zu beachten, daß dies in der Konfigurationsart virtueller Host nicht 100% logisch ist

```
<VirtualHost 192.0.2.1>
        ...
        Bind 2001:0DB8::1
        ...
</VirtualHost>
```

Mehr ist nicht zu tun.

22.9. Andere Daemons

Seit einiger Zeit ist dies meist einfach, suchen Sie einfach nach einer Kommandozeilen-Option oder einer Konfigurationsvariable, um das Lauschen an IPv6-Adressen zu aktivieren. Schauen Sie dazu in den Manual-Seiten des Daemons oder in den entsprechenden FAQs nach. Es kann allerdings durchaus sein, daß sich der Daemon nur an die IPv6-"any"-Adresse (::) binden läßt und kein dediziertes Binden an eine spezielle IPv6-Adresse möglich ist (das hängt von der Unterstützung des Programmierers ab).

Kapitel 23. Programmierung

23.1. Programmierung mit Nutzung der C-API

Dieser Abschnitt ist momentan nicht in Deutsch verfügbar, die englische Version ist verfügbar unter: TLDP / Linux+IPv6-HOWTO / Programming using C-API

23.2. Andere Programmiersprachen

23.2.1. JAVA

In Sun Java Versionen ab 1.4 ist Unterstützung für IPv6 vorhanden, siehe dazu auch die Klasse Inet6Address (1.5/5.0). Weitere Tipps sind verfügbar im *Networking IPv6 User Guide for JDK/JRE* 1.4 und 1.5 (5.0).

23.2.2. Perl

Stand Mai 2007 ist nichts bekannt, daß der Kern von Perl IPv6 nativ unterstützt. Dies kann allerdings aktiviert werden durch Benützung des folgenden Moduls:

- Socket6

Zudem existieren weitere Module für/mit IPv6 Unterstützung (z.B. Net::IP), suche nach "IPv6" bei http://search.cpan.org/.

Kapitel 24. Interoperabilität

Das TAHI Project prüft das Zusammenspiel der verschiedenen Betriebssysteme in Hinblick auf IPv6 Funktionalität und Implementierung. Der Linux Kernel hat bereits das IPv6 Ready Logo Phase 1 bekommen.

Kapitel 25. Weitere Informationen und URLs

25.1. Gedruckte Bücher, Artikel, Onlinerezensionen

25.1.1. Gedruckte Bücher (Englisch)

25.1.1.1. Cisco

- Cisco Self-Study: Implementing IPv6 Networks (IPV6), von Regis Desmeules. Cisco Press; ISBN 1587050862; 500 Seiten; 1. Edition (April 11, 2003). Anmerkung: Dieser Titel wird am 11. April 2003 publiziert.
- Configuring IPv6 with Cisco IOS, von Sam Brown, Sam Browne, Neal Chen, Robbie Harrell, Edgar, Jr. Parenti (Editor), Eric Knipp (Editor), Paul Fong (Editor) 362 Seiten; Syngress Media Inc; ISBN 1928994849; (July 12, 2002).

25.1.1.2. Allgemein

- IPv6 in Practice: A Unixer's Guide to the Next Generation Internet von Benedikt Stockebrand, November 2006; ISBN 3-540-24524-3
- IPv6 Essentials von Silvia Hagen, zweite Auflage, Mai 2006; ISBN 0-5961-0058-2 ToC, Index, Sample Chapter etc.; O'Reilly Pressrelease
- IPv6: The New Internet Protocol. Von Christian Huitema; Publiziert von Prentice-Hall; ISBN 0138505055. Beschreibung: Dieses Buch, geschrieben von Christian Huitema - einem Mitglied des Internet Architecture Board, bietet eine exzellente Beschreibung von IPv6, die Unterschiede zu IPv4 sowie die 'wies' und 'warums' der IPv6 Entwicklung. Quelle: http://www.cs.uu.nl/wais/html/na-dir/internet/tcp-ip/resource-list.html
- IPv6 Networks von Niles, Kitty; (ISBN 0070248079); 550 Seiten; Datum der Veröffentlichung: 05/01/1998.
- Implementing IPV6. Supporting the Next Generation Internet Protocols von P. E. Miller, Mark A. Miller; Hrsg.: John Wiley & Sons; ISBN 0764545892; 2. Edition (15.März 2000); 402 Seiten.
- Big Book of Ipv6 Addressing Rfcs von Peter H. Salus (Compiler), Morgan Kaufmann (Hrsg.), April 2000, 450 Seiten ISBN 0126167702.
- Understanding IPV6 von Davies, Joseph; ISBN 0735612455; Datum der Veröffentlichung: 05/01/2001; 350 Seiten.
- Migrating to IPv6 - IPv6 in Practice von Marc Blanchet; John Wiley & Sons (Hrsg.); ISBN 0471498920; 1. Edition (November 2002); 368 Seiten.
- Ipv6 Network Programming von Jun-ichiro Hagino; ISBN 1555583180
- Wireless boosting IPv6 von Carolyn Duffy Marsan, 10/23/2000.
- O'reilly Network search for keyword IPv6 ergibt 29 Treffer (28. Januar 2002)

25.1.2. Artikel, eBooks, Online Rezensionen

- Getting Connected with 6to4 von Huber Feyrer, 06/01/2001
- Transient Addressing for Related Processes: Improved Firewalling by Using IPv6 and Multiple Addresses per Host; geschrieben von Peter M. Gleiz, Steven M. Bellovin (PC-PDF-Version; Palm-PDF-Version; PDB-Version)
- Internetworking IPv6 with Cisco Routers von Silvano Gai, McGrawHill Italia, 1997. Die 13 Kapitel und der Anhang A-D sind als PDF-Dokument 'downladbar'.
- Migration and Co-existence of IPv4 and IPv6 in Residential Networks von Pekka Savola, CSC/FUNET, 2002

25.1.3. Wissenschaftliche Publikationen (Kurzbeschreibungen, Bibliographien, Online Quellen)

Siehe auch: liinwww.ira.uka.de/ipv6 bzw. Google / Scholar / IPv6

- GEANT IPv6 Workplan
- IPv6 Trials on UK Academic Networks: Bermuda Project Aug.2002: Participants - Getting connected - Project deliverables - Network topology - Address assignments - Wireless IPv6 access - IPv6 migration - Project presentations - Internet 2 - Other IPv6 projects - IPv6 fora and standards Bermuda 2...
- http://www.ipv6.ac.uk/
- IPv6 at the University of Southampton
- Microsoft Research IPv6 Implementation (MSRIPv6): MSRIPv6 Configuring 6to4 - Connectivity with MSR IPv6 - Our 6Bone Node...

25.1.4. Sonstiges

Mehr Infos gibt es unter: SWITCH IPv6 Pilot / References

25.2. Konferenzen und Meetings

Fehlt etwas? Vorschläge sind willkommen!

25.2.1. 2004

- 1st Global IPv6 Summit in Sao Paul, Brasil

25.3. Online-Informationen

25.3.1. Mit dem IPv6 Backbone verbinden

Mehr Infos in späteren Versionen... Vorschläge sind Willkommen!

25.3.1.1. Globale Registrierungsstellen

Siehe regionale Registrierungsstellen.

25.3.1.2. Regionale Haupt-Registrierungsstellen

- Amerika: ARIN, ARIN / registration page, ARIN / IPv6 guidelines
- EMEA: Ripe NCC, Ripe NCC / registration page, Ripe NCC / IPv6 registration
- Asien/Pazifik: APNIC, APNIC / IPv6 ressource guide
- Latein Amerika und Karikik: LACNIC, IPv6 Registration Services, IPv6 Allocation Policy
- Afrika: AfriNIC

Es existiert auch eine Liste der größten Zuteilungen sortiert nach lokalen Registrierungsstellen: Ripe NCC / IPv6 allocations.

25.3.1.3. Tunnel-Broker

Anmerkung: Eine Tunnel-Broker Liste ist im Abschnitt Tunnel broker weiter unten zu finden.

- Former IPng. Tunnelbroker and IPv6 resources, now migrated to the SixXs System.
- Eckes IPv6-with-Linux Seite.
- tunnelc - ein Perl basiertes Tunnel Client Script: freshmeat.net: Project details for tunnel client SourceForge: Project Info - tunnelc (also here)
- Linux Advanced Routing & Traffic Control HOWTO, Chapter 6: IPv6 tunneling with Cisco and/or 6bone.

25.3.1.4. 6to4

- NSayer's 6to4 information
- RFC 3068 / An Anycast Prefix for 6to4 Relay Routers

25.3.1.5. ISATAP

- ISATAP (Intra-Site Automatic Tunnel Access Protocol) Information by JOIN

25.3.2. Neueste Nachrichten und URLs zu anderen Dokumenten

- Viele URLs zu anderen Dokumenten von Anil Edathara
- go6 - The IPv6 Portal: ein IPv6 Online-Portal mit einem WIKI-basierenden IPv6-Know-How-Schwerpunkt, einem IPv6-Diskussionsforum, einer aktuellen Sammlung von IPv6-Events und Nachrichten, freiem IPv6-Zugang und Services, IPv6 Software-Applikationen und viel mehr

---

25.3.3. Protokoll-Informationen

25.3.3.1. IPv6 bezogene Request For Comments (RFCs)

Das veröffentlichen einer Liste mit IPv6 relevanter RFCs geht über den Rahmen dieses Dokumentes hinaus, unter folgenden Links können Sie jedenfalls diverse Listen finden:

- sortierte Liste:IPng Standardization Status oder IPng Current Specifications von Robert Hinden
- IPv6 Related Specifications auf IPv6.org

25.3.3.2. Aktuelle Entwürfe diverser Arbeitsgruppen

Aktuelle (auch) IPv6-bezogene Drafts finden Sie hier:

- IP Version 6 (ipv6)
- Next Generation Transition (ngtrans)
- Dynamic Host Configuration (dhc)
- Domain Name System Extension (dnsext)
- IPv6 Operations (v6ops)
- Mobile IP (mobileip)
- Get any information about IPv6, from overviews, through RFCs & drafts, to implementations (inklusive Verfügbarkeit der Stacks auf verschiedenen Plattformen & Quellcode diverser IPv6 Stacks)

25.3.3.3. Sonstige

- SWITCH IPv6 Pilot / References, umfangreiche Liste mit IPv6 Quellen betreut von Simon Leinen

25.3.4. Weitere Informationen

DeepSpace6 / Weitere interessante Links

25.3.4.1. Linux Informationen

- DeepSpace6 / (Not only) Linux IPv6 Portal - Italien (Spiegel)
- IPv6-HowTo for Linux by Peter Bieringer - Deutschland, und sein Bieringer / IPv6 - software archive
- Linux+IPv6 status by Peter Bieringer - Deutschland
- DeepSpace6 / IPv6 Status Page - Italien (Spiegel) (ersetzt das oben genannte in Zukunft)
- USAGI project - Japan, und deren USAGI project - software archive
- Linux Optimized Link State Routing Protocol (OLSR) IPv6 HOWTO
- LinShim6

25.3.4.2. Informationen zu Linux-Distributionen

PLD
: PLD Linux Distribution ("Marktführer" bei inkludierten IPv6 fähigen Paketen))

Red Hat
: Red Hat Enterprise Linux, Pekka Savola's IPv6 packages

Fedora
: Fedora Core Linux

Debian
: Debian Linux, IPv6 with Debian Linux

Novell/SuSE
: Novell/SuSE Linux

Mandriva
: Mandriva

Weitere Details siehe unter IPv6+Linux Status Distributions.

25.3.4.3. Allgemeine Informationen

- IPv6.org
- 6bone
- WIDE project - Japan
- SWITCH IPv6 Pilot - Schweiz
- IPv6 Corner of Hubert Feyrer - Deutschland
- IPv6 Forum - ein weltweites Konsortium führender Internet-Hersteller, Forschungs- & Bildungseinrichtungen...
- Playground.sun.com / IPv6 Info Page - betreut von Robert Hinden, Nokia. Hier gibt es jede Information zum Thema IPv6: Zusammenfassungen, RFCs & Drafts, Implementierungen (Verfügbarkeit der Stacks auf verschiedenen Plattformen & Quellcode diverser IPv6 Stacks).
- 6INIT - IPv6 Internet Initiative - ein Fifth Framework Projekt der EU im Rahmen des IST Programmes.
- IPv6 Task Force (European Union)
- 6init - IPv6 INternet IniTiative

- IPv6: The New Version of the Internet Protocol, von Steve Deering.
- IPv6: The Next Generation Internet Protocol, von Gary C. Kessler.
- IPv6: Next Generation Internet Protocol - 3Com
- internet || site und internet2 Working Group
- NetworkWorldFusion: Search / Doc Finder: searched for IPv6 (102 Dokumente gefunden - 22.12.2002)
- The Register (Suche nach IPv6 ergab 30 Dokumente, 22.12.2002)
- ZDNet Search for IPv6
- TechTarget Search for IPv6
- IPv6 & TCP Resources List

Fehlt etwas? Vorschläge sind Willkommen!

25.3.4.4. Marktforschung

- A Tale of Two Wireless Technology Trends: Processor Development Outsourcing and IPv6Yankee Group - 4/1/2002 - 12 Seiten - ID: YANL768881
- The World Atlas of the Internet: Americas; IDATE - 2/1/2002 - 242 Seiten - ID: IDT803907. Folgende Länder werden behandelt: Zentralamerika, Nordamerika,Südamerika; Listenpreis: $ 3,500.00; Exzerpt: Panorama of Internet access markets across the globe. Market assessment and forecasts up to 2006 for 34 countries: market structure: main ISPs and market shares; number of subscribers, of ISPs.
- Early Interest Rising for IPv6 von IDC (Autor); Listenpreis: $1,500.00; Edition: e-book (Acrobat Reader); Hrsg.: IDC; ISBN B000065T8E; (1. March 2002)

25.3.4.5. Patente

- Delphion Research: Patent Search Page. Basic (kostenlose) Registrierung ist notwendig. Beispiele für die Suche nach IPv6 (vom 21.12.2002): Communicating method between IPv4 terminal and IPv6 terminal and IPv4-IPv6 converting apparatus Translator for IP networks, network system using the translator, and IP network coupling method therefor

---

25.3.5. Sortiert nach Ländern

25.3.5.1. Europa

- www.ist-ipv6.org: IST IPv6 Cluster, European IPv6 Research and Development Projects
- Euro6IX: European IPv6 Internet Exchanges Backbone

25.3.5.2. Australien

- Carl's Australian IPv6 Pages (alter Inhalt)

25.3.5.3. Belgien

Vorschläge sind Willkommen!

25.3.5.4. Brasilien

- IPv6 do Brasil

25.3.5.5. China

Vorschläge sind Willkommen!

25.3.5.6. Deutschland

- OpenBC / IPv6

25.3.5.7. Frankreich

- Renater: Renater IPv6 Projekt Seite
- IPv6 - RSVP - ATM at INRIA
- NetBSD IPv6 Dokumentation

25.3.5.8. Großbritannien

- British Telecom IPv6 Home: BT's ISP IPv6 Versuch, Englands erster IPv6 Internet Exchange etc.

25.3.5.9. Indien

Vorschläge sind willkommen!

25.3.5.10. Italien

- Project6: IPv6 mit Linux

25.3.5.11. Japan

- Yamaha IPv6 (sorry, alles in japanischer Sprache ...)

25.3.5.12. Korea

- ETRI: Electronics and Telecommunications Research Institut
- IPv6 Forum Korea: IPv6 Infrastruktur Projekt in Korea

25.3.5.13. Mexiko

- IPv6 Mexico (spanische & englische Version) - IPv6 Projekt Homepage der National Autonomous University of Mexico (UNAM)

25.3.5.14. Niederlande

- SURFnet: SURFnet IPv6 Backbone
- STACK, STACK (IPv6): Computer-Studendten-Verband der Eindhoven University of Technology, Niederlande.
- IPng.nl: Zusammenarbeit zwischen WiseGuys und Intouch.

25.3.5.15. Österreich

- IPv6@IKNnet and MIPv6 Research Group: TU Vienna, Austria (IPv6: Projekte, Publikationen, Diplom- / Doktorarbeiten, Konferenzunterlagen etc.)

25.3.5.16. Portugal

Vorschläge sind willkommen!

25.3.5.17. Russland

- IPv6 Forum for Russia: Yaroslavl State University Internet Center

---

25.3.5.18. Schweiz

Vorschläge sind willkommen!

25.3.6. Sortiert nach Betriebssystemen

25.3.6.1. *BSD

- KAME project (*BSD)
- NetBSD's IPv6 Networking FAQ
- FreeBSD Ports: Ipv6

25.3.6.2. Cisco IOS

- Cisco IOS IPv6 Entry Page
- IPv6 for Cisco IOS Software, Datei 2 von 3: Aug 2002 -- Inhalt: IPv6 for Cisco IOS Software; Configuring Documentation Specifics; Enabling IPv6 Routing and Configuring; IPv6 Addressing; Enabling IPv6 Processing Globally.
- Cisco Internet Networking Handbook, Chapter IPv6

25.3.6.3. HPUX

- comp.sys.hp.hpux FAQ

25.3.6.4. IBM

- Now that IBM's announced the availability of z/OS V1.4, what's new in this release? Die Frage wurde am 15. August 2002 'geposted'.

25.3.6.5. Microsoft

- Microsoft Windows 2000 IPv6
- MSRIPv6 - Microsoft Research Network - IPv6 Homepage
- Internet Connection Firewall Does Not Block Internet Protocol Version 6 Traffic (6.11.2001)
- Internet Protocol Numbers (8.10.2002)
- IPv6 Technology Preview Refresh (16.10.2002)
- HOW TO: Install and Configure IP Version 6 in Windows .NET Enterprise Server (26.10.2002)

- Windows .NET Server 6to4 Router Service Quits When You Advertise a 2002 Address on the Public Interface (28.10.2002)
- msdn - Microsoft Windows CE .NET - IPv6 commands

25.3.6.6. Solaris

- Sun Microsystems Solaris
- Solaris 2 Frequently Asked Questions (FAQ) 1.73

25.3.6.7. Sumitoma

- Sumitomo Electric has implemented IPv6 on Suminet 3700 family routers

---

25.3.6.8. ZebOS

- IpInfusion's ZebOS Server Routing Software

25.3.7. IPv6 Sicherheit

- Internet Security Systems: Security Center, X-Force Database Search (21.12.2002 - 6 Themen bez. IPv6 gefunden)
- NIST IPsec Project (National Institute of Standards and Technology, NIST)
- Information Security
- NewOrder.box.sk (search for IPv6) (Artikel, Exploits, Datei-Datenbank etc.)

25.3.8. Programm-Listen

- DeepSpace6 / IPv6 Status Page (Mirror)
- IPv6.org / IPv6 enabled applications
- Freshmeat / IPv6 search, aktuell (14 Dez. 2002) 62 Projekte
- IPv6 Forum / Web Links

25.3.8.1. Analyse-Werkzeuge

- Wireshark (ehemals *Ethereal*) is ein kostenloser Netzwerkprotokoll-Analyseprogramm für Unix und Windows
- Radcom RC100-WL - Download Radcom RC100-WL Protokollanalyseprogramm version 3.20

25.3.8.2. IPv6 Produkte

- 6wind - Lösungen für IPv4/IPv6 Router, QoS, Multicast, Mobility, Security/VPN/Firewall.
- Fefe's patches for IPv6 with djbdns - Aug 2002 - Was ist djbdns und warum es IPv6 benötigt? djbdns ist ein vollwertiger DNS Server, welcher "outperforms BIND in nearly all respects".
- ZebOS Server Routing Suite
- SPA Mail Server 2.21
- Inframail (Advantage Server Edition) 6.0
- HTTrack Website Copier
- CommView 5.0
- Posadis 0.50.6

25.3.8.3. SNMP

- comp.protocpols.snmp SNMP FAQ Part 1 of 2

25.4. IPv6 Infrastruktur

25.4.1. Statistiken

- IPv6 routing table history erstellt von Gert Döring, Space.Net
- Official 6bone Webserver list Statisic

25.4.2. Internet Exchanges

Eine weitere Liste von IPv6 Internet Exchanges gibt es unter: IPv6 status of IXPs in Europe

25.4.2.1. Deutschland

- INXS: (Cable & Wireless) München und Hamburg

25.4.2.2. Estlanda

- TIX (Tallinn Interneti eXchange mit IPv6 Support)

25.4.2.3. Europa

- Euro6IX, European IPv6 Internet Exchange Backbone

25.4.2.4. Frankreich

- French National Internet Exchange IPv6 (seit 1.11.2002 aktiv). FNIX6 bietet ISPs im Großraum Paris den Dienst eines kostenlosen und zuverlässlichen High Speed FastEthernet Internet Exchange.

25.4.2.5. Großbritannien

- UK6X: London
- XchangePoint: London

25.4.2.6. Japan

- NSPIXP-6: IPv6--basierter Internet Exchange in Tokio
- JPIX, Tokio

25.4.2.7. Korea

- 6NGIX

25.4.2.8. Niederlande

- AMS-IX: Amsterdam Internet Exchange

25.4.2.9. USA

- 6TAP: Chicago. Bietet weltweit Peerings an.
- PAIX: Palo Alto

25.4.3. Tunnel broker

Auch interessant: http://www.deepspace6.net/docs/tunnelbrokers.html

25.4.3.1. Belgien

Fehlt etwas? Vorschläge sind willkommen!

25.4.3.2. Canada

- Freenet6 - /48 Delegation, Canada Getting IPv6 Using Freenet6 on Debian Freenet6 creater

25.4.3.3. China

Fehlt etwas? Vorschläge sind willkommen!

25.4.3.4. Deutschland

- 6bone Knoten Leipzig Info bez. Hackangriff (2001)

25.4.3.5. Estlanda

- Estpak

25.4.3.6. Großbritannien

- NTT, Großbritannien - IPv6 Versuch. IPv4 Tunnel und native IPv6 Standleitungs-Verbindungen. POPs gibt es in: London, Düsseldorf, New Jersey (USA, East Coast) Cupertino (USA, West Coast) Tokio

25.4.3.7. Italien

- Comv6
- Bersafe (italienische Sprache)

25.4.3.8. Japan

Fehlt etwas? Vorschläge sind willkommen!

25.4.3.9. Malaysia

Fehlt etwas? Vorschläge sind willkommen!

25.4.3.10. Niederlande

- IPng Netherland - Intouch, SurfNet, AMS-IX, UUNet, Cistron, RIPE NCC und AT&T sind am AMS-IX angeschlossen. Unter bestimmten Voraussetzungen ist es möglich, einen statischen Tunnel zu bekommen.
- SURFnet Customers

25.4.3.11. Norwegen

- UNINETT - Pilot IPv6 Service (für Kunden): tunnelbroker & address allocation Uninett-Autoupdate-HOWTO

25.4.3.12. Spanien

- Consulintel

25.4.3.13. Schweiz

Fehlt etwas? Vorschläge sind willkommen!

25.4.3.14. USA

- ESnet, USA - Energy Sciences Network: Tunnel Registry & Address Delegation für direkt angeschlossene ESnet Sites und ESnet Partner.
- Hurricane Electric, US backbone; Hurrican Electric Tunnelbroker (also available under http://tunnelbroker.com/) Presseaussendung: Hurricane Electric Upgrades IPv6 Tunnel Broker Tunnel Broker Endpoint Autoupdate, Perl Skript

---

25.4.3.15. Singapore

Fehlt etwas? Vorschläge sind Willkommen!

25.4.3.16. Weitere Tunnel broker...

- Public 6to4 relay routers (MS IIE Boycott!)

25.4.4. Native IPv6 Dienste

Anmerkung: Die folgenden Dienste sind meist nur mit einer gültigen IPv6 Verbbindung erreichbar!

25.4.4.1. Net News (NNTP)

Fehlt etwas? Vorschläge sind Willkommen!

25.4.4.2. Spiele Server

- Quake2 über IPv6

25.4.4.3. IRC Server

Fehlt etwas? Vorschläge sind Willkommen!

25.4.4.4. Radiosender, Musik-Streams

Fehlt etwas? Vorschläge sind Willkommen!

25.4.4.5. Web Server

- Peter Bieringer's Home of Linux IPv6 HOWTO

Fehlt etwas? Vorschläge sind Willkommen!

25.5. Mailinglisten

Weitere Listen von Mailinglisten sind verfügbar unter:

- DeepSpace6 / Mailling Lists

Die größten Mailinglisten sind in folgender Tabelle zusammengefasst:

.

Schwerpunkt	Request e-mail Adresse	Abonnieren	e-mail Adresse der Mailingliste	Sprache	Zugang via WWW
Linux Kernel Networking inkl. IPv6	majordomo (at) vger.kernel.org	netdev	netdev (at) vger.kernel.org	Englisch	Info, Archive
Mobile IP(v6) für Linux	Web-based, see URL	mipl	mipl (at) mobile-ipv6.org	Englisch	Info, Archive
Linux IPv6 User & USAGI	usagi-users-ctl (at) linux-ipv6.org		usagi-users (at) linux-ipv6.org	Englisch	Info / Search, Archive
IPv6 und Debian Linux	Web-based, siehe URL		debian-ipv6 (at) lists.debian.org	Englisch	Info/Subscription/Archive
6bone	majordomo (at) isi.edu	6bone	6bone (at) isi.edu	Englisch	Info, Archive
IPv6 User allgemein	majordomo (at) ipv6.org	users	users (at) ipv6.org	Englisch	Info, Archive
Bugtracking Internet Programme (1)	bugtraq-subscribe (at) securityfocus.com		bugtraq (at) securityfocus.com (3)	Englisch	Info, Archive

.

(1) sehr empfohlen wenn Sie Server-Programme zur Verfügung stellen.

(2) Mailingliste ist moderiert.

Fehlt etwas? Vorschläge sind Willkommen!

Via Web sind auch folgende Mailinglisten & Newsgroups verfügbar:

- student-ipv6 (India) Beschreibung: Dies ist eine Newsgruppe für die 'Student Awareness group of IPv6' in Indien
- sun-ipv6-users Beschreibung: Bitte berichten Sie Probleme/Vorschläge in Bezug auf die IPng Implementation von SUN Microsystems
- IPv6-BITS Beschreibung: Diese Liste dient der Koordination des Projekts Vertebrae.
- linux-bangalore-ipv6 Beschreibung: Die IPv6 deployment Liste der Bangalore Linux User Group
- packet-switching Beschreibung: Dies Liste behandelt folgende Themen: packet switching theory, technology, implementation and application in any relevant aspect including without limitation LAPB, X.25, SDLC, P802.1d, LLC, IP, IPv6, IPX, DECNET, APPLETALK, FR, PPP, IP Telephony, LAN PBX systems, management protocols like SNMP, e-mail, network transparent window systems, protocol implementation, protocol verification, conformance testing and tools used in maintaining or developing packet switching systems.
- de.comm.protocols.tcp-ip Beschreibung: Umstellung auf IPv6 Quelle: Chartas der Newsgruppen in de.*
- Google Group: comp.protocols.tcp-ip
- Google Group: linux.debian.maint.ipv6
- Google Group: microsoft.public.platformsdk.networking.ipv6
- Google Group: fa.openbsd.ipv6

25.6. Online-Werkzeuge

25.6.1. Test-Werkzeuge

- ping, traceroute, tracepath, 6bone registry, DNS: JOIN / Testtools (nur in deutscher Sprache, sollte aber keine Probleme für nicht Deutsch sprechende Personen sein)
- traceroute6, whois: IPng.nl
- AAAA Lookup Checker http://www.cnri.dit.ie/cgi-bin/check_aaaa.pl

25.6.2. Informationsbeschaffung

- List of worldwide all IPv6-aggregated IP-Blocks

25.6.3. IPv6 Looking Glasses

- DRENv6 Looking Glass

25.6.4. Hilfsapplikationen

- IPv6 Prefix Calculator von TDOI
- DNS record checker

25.7. Trainings, Seminare

- CIW Internetworking Professional Training CBT CD
- Training Pages, U.K. - Suche nach IPv6 (8 Kurse, 2006-08-21)
- Erion IPv6 Training. UK

Fehlt etwas? Vorschläge sind willkommen!

25.8. 'Die Online Entdeckung' ...

IPv6: Addressing The Needs Of the Future von Yankee Group (Autor)

Listenpreis: $595.00

Edition: e-book (Acrobat Reader)

Seiten: 3 (drei)

Hrsg.: MarketResearch.com; ISBN B00006334Y; (1. November 2001)

;-) Die Auflagenhöhe dieses eBooks wäre doch sehr interessant...

Kapitel 26. Versions-Überblick / Danksagung / Zum Schluss

26.1. Versions-Überblick

Die Versionen x.y. werden im Internet veröffentlicht.

Die Versionen x.y.z stellen Zwischenschritte dar, die nur als LyX- und SGML-Datei im TLDP-CVS veröffentlicht werden. Dadurch, dass allerdings Deep Space 6 diese SGML-Dateien spiegelt und daraus öffentliche Versionen generiert, sind diese auch dort und auf den Servern, die Deep Space 6 spiegeln, verfügbar.

26.1.1. Ausgabe 0.x

26.1.1.1. Englische Sprachversion (Peter Bieringer's Original)

Die Historie der Änderungen der englischen Sprachversion finden Sie dort: TLDP / Linux+IPv6-HOWTO / Revision History.

26.1.1.2. Deutsche Sprachversion

0.63.de.1
: 2009-02-14/PB: Sync mit Original

0.62.de.1
: 2008-11-09/PB: Sync mit Original

0.61.1.de.3
: 2007-08-25/PB: Fix Typo

0.61.1.de.2
: 2007-08-17/PB: Fix Typo

0.61.1.de.1
: 2007-11-11/PB: Sync mit Original

0.61.de.1
: 2007-10-06/PB: Sync mit Original, kleine Korrekturen

0.60.de.1
: 2007-05-31/PB: Sync mit Original (bis auf C-API), kleine Korrekturen

0.51.de.1
: 2006-11-08/PB: Sync mit Original

0.50.2.de.1
: 2006-10-25/PB: Sync mit Original

0.50.1.de.1
: 2006-09-23/PB: Sync mit Original

0.50.de.1
: 2006-08-24/PB: Sync mit Original

0.49.5.de.1
: 2006-08-23/PB: Sync mit Original

0.49.4.de.1
: 2006-08-21/PB: Sync mit Original, Korrektur bei Adresstypen (Übersetzungsfehler)

0.49.3.de.1
: 2006-08-20/PB: Sync mit Original

0.49.2.de.1
: 2006-08-20/PB: Sync mit Original

0.49.1.de.1
: 2006-06-13/PB: Sync mit Original

0.49.de.1
: 2005-10-03/PB: Sync mit Original

0.48.de.1
: 2005-01-11/PB: Sync mit Original

0.47.de.1
: 2004-08-30/PB: Sync mit Original

0.46.5.de.1
: 2004-07-22/PB: Sync mit Original

0.46.4.de.1
: 2004-07-19/PB: Sync mit Original und kleine Korrekturen der Übersetzung

0.46.2.de.1
: 2004-05-22/PB: Sync mit Original

0.46.1.de.1
: 2004-04-18/PB: Sync mit Original

0.46.de.1
: 2004-04-04/PB: Sync mit Original (Italienische Übersetzung verfügbar, Informationen über DHCPv6, kleinere Updates)

0.45.1.de.1
: 2004-01-12/PB: Sync mit Original (kleine Erweiterung)

0.45.de.1
: 2004-01-11/PB: Sync mit Original (kleine Korrekturen, URL-Korrekturen und Erweiterungen)

0.44.2.de.1
: 2003-10-30/PB: Sync mit Original (kleine Korrektur)

0.44.de.1
: 2003-08-15/PB: Sync mit Original (URL-Korrekturen, Tipp bei tcp_wrappers und Apache2)

0.43.2.de.2
: 2003-07-26/PB: Sync mit Original (URL-Korrekturen)

0.43.2.de.1
: 2003-06-11/PB: Sync mit Original (URL-Korrekturen)

0.43.1.de.2
: 2003-06-11/PB: Typo gefixt

0.43.1.de.1
: 2003-06-07/PB: Sync mit Original (Links, IPsec)

0.43.de.1
: 2003-06-05/PB: Sync mit Original (SuSe-Linux-Tipps)

0.41.4.de.1
: 2003-05-02/PB: Sync mit Original (URL-Korrekturen)

0.41.3.de.1
: 2003-04-23/PB: Sync mit Original (URL-Korrektur)

0.41.1.de.1
: 2003-03-31/PB: Sync mit Original (URL-Korrektur)

0.41.de.1
: 2003-03-22/PB: Angabe der URL dieser Übersetzung

0.40.2.de.1
: 2003-02-27/PB: Sync mit Original (URL-Korrektur)

0.40.1.de.1

2003-02-12/PB: Sync mit Original (Debian-Linux-Konfiguration), Ersetzen der Info über verfügbare Übersetzungen durch einen Link zum Original.

0.40.de.1

2003-02-10/PB: Kleine sprachliche Änderungen und paar kleine Verbesserungen.

0.39.2.de.1

2003-02-10/GK: Erste öffentliche Version in deutscher Sprache. Diese Version entspricht inhaltlich der englischen Version 0.39.2. Der Zusatz de.1 bezeichnet die Revision der deutschen Sprachversion. Diese Version wurde unter Mandrake Linux 9.0 mit LyX Version 1.2.1 und der Klasse DocBook book (SGML) erstellt.

26.2. Danksagung

Dieser feinen Liste hinzugefügt werden können Sie am schnellsten, indem Sie mir Bug fixes, Korrekturen und/oder Updates schicken ;-)

Wenn Sie eine größere Überarbeitung vornehmen wollen, können Sie dazu die LyX Datei (siehe original source) verwenden und mir entsprechende diffs dann zusenden, diffs der SGML-Version sind hingegen nicht sehr nützlich.

26.2.1. Primärer Dank...

- David Ranch <dranch at trinnet dot net>: Er ermutigte mich zum Schreiben dieses HOWTOs, für seine Anmerkungen zu den ersten englischen Versionen, für seine Beiträge zu den IPv6 Test-Ergebnissen auf der IPv6-Homepage des Autors. Ebenfalls für seine größeren Überarbeitungen und Vorschläge.
- Pekka Savola <pekkas at netcore dot fi>: Für größere Überarbeitungen der englischen Version, seinen Input und Vorschläge.
- Georg Käfer <gkaefer at gmx dot at>: Für das Aufspüren eines Fehlers bei der PDF Erstellung (durch LDP Betreuer Greg Ferguson behoben), den Input betreff deutscher Bücher, einer großen URL-Liste, die Überprüfung aller Links und vieler weiterer Vorschläge, Korrekturen und Beiträge.

26.2.2. Sonstiger Dank...

26.2.2.1. Verwaltung des Dokuments

Als Neuling ein LDP HOWTO zu schreiben (in LyX schreiben sowie SGML konformer Export zu DocBook) ist nicht so einfach, wie von so manchem behauptet wird. Es gibt einige sonderbare Fallen... Nichtsdestoweniger Dank an:

- Autoren des LDP Author Guide
- B. Guillon: Für sein DocBook with LyX HOWTO

26.2.2.2. Inhalt des Dokuments

Mein Dank für Fixes und Hinweise ist hier aufgelistet - und die Liste wird mit der Zeit sicherlich länger werden... Bezüglich der englischen Version finden Sie in dieser, hier werden nur diejenigen, die die deutsche Version betreffen, aufgelistet.

- Nico Schottelius, Oliver Kraus, Götz Bauermann: Typo gefunden
- Jens Nachtigall: Fehler gefunden

26.3. Zum Schluss

Danke für's Lesen. Hoffentlich ist es von Nutzen!

Falls Sie irgendeine Frage haben, abonnieren sie eine entsprechende maillist und diskutieren Sie ihr Problem unter Angabe möglichst vieler Informationen.