Aufbau eines Linux-basierten IPv6-Routers zum
Einsatz im Umfeld eines Internet-Service-Providers

Diplomarbeit

zur Erlangung des akademischen Grades
Diplomingenieur (FH)
an der Fachhochschule für Technik und Wirtschaft Berlin

Fachbereich Ingenieurwissenschaften I
Studiengang Technische Informatik

Betreuer: Prof. Dr. Johann Schmidek
Eingereicht von: André Grüneberg

Blankenfelde, 23. Juni 2004

Vorwort

Im Jahr 1999 hörte ich auf dem *16. Chaos Communication Congress* einen Vortrag von Felix von Leitner zum Thema IPv6. Ich war sofort von den Möglichkeiten begeistert, die das neue Internet-Protokoll bieten sollte. Insbesondere der Kommentar, dass es einen *Hack Value* habe, veranlasste mich zu weiteren Erkundungen.

Ich stellte schnell fest, dass bis zu einem Durchbruch von IPv6 noch viel zu tun sein würde. Vor allen Dingen schienen die Router-Hersteller mit den größten Marktanteilen nur sehr langsam zu reagieren. Auch durch den Druck der Politik werden heute zwar IPv6-fähige Router angeboten, die Aufrüstung vorhandener Netzwerke vollzieht sich jedoch nur langsam. Die Tatsache, dass die Betriebssoftware der Router noch nicht ausgereift ist, scheint dabei einer der wichtigsten Gründe zu sein.

Gleichzeitig machte die Leistungsfähigkeit moderner PCs gewaltige Fortschritte. Mit den freien Betriebssystemen, die seit mehreren Jahren IPv6 unterstützen, steht somit eine gute Basis für einen IPv6-Router zur Verfügung. Mit Hilfe von *Linux* und weiterer freier Software erschien es mir möglich, einen Router zu entwickeln, der IPv6 unterstützt und den Bedürfnissen eines Internet-Service-Providers gerecht wird.

Nach diversen Schwierigkeiten fand ich Mitte 2003 – ca. zehn Jahre nach Beginn der Entwicklungen um IPv6 – endlich einen Zugangs-Provider, der mir eine native IPv6-Anbindung für einen *T-DSL*-Anschluss anbieten konnte – die *Logivision GmbH* in Berlin.

Der *Logivision GmbH* und insbesondere Herrn Dipl.-Ing. Alfred Schweder ist die Inspiration zu dieser Arbeit zu verdanken. Vor allen Dingen möchte ich mich hiermit für die Einblicke in die Technik eines Internet-Service-Providers und die ständige Hilfsbereitschaft bedanken.

Weiteren Dank möchte ich meinen Freunden, meinen Eltern und meiner Schwester aussprechen. Sie alle haben mich in den letzten Wochen mit viel Geduld begleitet und verschiedene Versionen dieser Arbeit gelesen, korrigiert und kommentiert.

Mein Dank gilt nicht zuletzt auch den enthusiastischen Mitgliedern der IPv6-Gemeinde, die seit vielen Jahren hartnäckig daran arbeiten, dass IPv6 ein Erfolg wird.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Quellcodeverzeichnis

Abkürzungsverzeichnis

ADSL	Asynchronous Digital Subscriber Line – digitale breitbandige Leitungstechnologie zum Anschluss von End-Kunden
AH	Authentication Header – dient zur Absicherung der IP-Pakete gegen Veränderung und Fälschung
AS	Autonomous System – Identifizierungs-Code eines Netzwerks im Inter-Domain-Routing
ASIC	Application Specific Integrated Circuit – Schaltkreis mit speziellen auf die Anwendung abgestimmten Funktionen
ATM	Asynchronous Transfer Mode – Netzwerk-Technologie, die auf Zell-Switching basiert und virtuelle Kanäle bereitstellt
BGP	Border Gateway Protocol – das am weitesten verbreitete Protokoll im Inter-Domain-Routing
CIDR	Classless Interdomain Routing – Konzept zur Überwindung der Klassen-Einteilung der IPv4-Adressen im Routing.
CLI	Command Line Interface – dient der Konfiguration mittels Kommandozeile
DAD	Duplicate Address Detection – Verfahren zur Prüfung der Einmaligkeit einer IPv6-Adresse
DoS	Denial of Service – Angriff auf ein System mit dem Ziel der Dienst-Unterbrechung
DFZ	Default Free Zone – Menge der Router, die keine Default-Route für die Routing-Entscheidung benutzen
DHCPv6	Dynamic Host Configuration Protocol für IPv6 – dient der automatischen Konfiguration verschiedener Netzwerk-Parameter
eBGP	Exterior Border Gateway Protocol – BGP-Verbindung zwischen zwei AS
EGP	Exterior Gateway Protocol – Protokolle des Inter-Domain-Routings
ESP	Encapsulating Security Payload – dient der Absicherung von IP-Paketen vor Veränderung, Fälschung und Mitlesen

iBGP	Interior Border Gateway Protocol – BGP-Verbindung zwischen Routern des gleichen AS
ICMPv6	Internet Control Message Protocol für IPv6 – Protokoll zur Übermittlung von Steuer-Informationen
IESG	Internet Engineering Steering Group
IETF	Internet Engineering Task Force
IGP	Interior Gateway Protocol – Routing-Protokoll das die Routen innerhalb einer Organisation verwaltet
IKE	Internet Key Exchange – Protokoll zur Vereinbarung von symmetrischen Schlüsseln für IPSec
IPSec	IP Security – Rahmen zur Absicherung von IP-Paketen
IPv6	Internet Protocol Version 6, ehem. IPng – IP Next Generation
ISP	Internet-Service-Provider
L2TP	Layer Two Tunneling Protocol – Verfahren zum Tunneln von PPP-Verbindungen über paketvermittelnde Netzwerke
LAC	L2TP Access Concentrator – Quelle einer L2TP-Verbindung
LDP	Label Distribution Protocol – Protokoll zum Austausch der für MPLS nötigen Label zwischen beteiligten Routern
LNS	L2TP Network Server – Ziel einer L2TP-Verbindung
MIB	Management Information Base – Definition des Datenmodells z. B. von SNMP
MPLS	Multi Protocol Label Switching – Verfahren zur Erhöhung der Effizienz beim Routing
MTU	Maximum Transfer Unit – maximale Anzahl Bytes, aus denen ein zu transportierendes Schicht-3-Paket bestehen darf
NAT	Network Address Translation – System zum Umschreiben von IP-Adressen beim Übergang in ein anderes Netzwerk, z. B. weil nur eine öffentliche IP-Adresse zur Verfügung steht
ND	Neighbor Discovery – auf ICMPv6 basierendes Protokoll zwischen Systemen, die an ein gemeinsames Netzwerk-Segment angeschlossen sind
NIC	Network Interface Card – Steckkarte die eine physische Verbindung zu einem Netzwerk herstellt
NTP	Network Time Protocol – dient der Synchronisation der Systemzeit per Netzwerk
OID	Object-ID

OSI-RM	OSI-Referenz-Modell
OSPF	Open Shortest Path First – ein IGP, das nach dem Link-State-Prinzip arbeitet
OSS	Open Source Software – Software deren Quellcode offengelegt ist und in veränderter Form weitergegeben werden darf
PIM	Protocol Independent Multicast – ein Multicast-Routing-Protokoll
PMTUD	Path MTU Discovery – Verfahren zur Erkennung der niedrigsten MTU auf dem Weg zu einem Ziel
PPP	Point-to-Point Protocol
pps	Packets per Second – Anzahl der pro Sekunde weitergeleiteten Pakete
RADIUS	Remote Authentication Dial-In User Service – Dienst zur zentralen Verwaltung von Authentifizierungs- und Accounting-Daten
RFC	Request for Comments – dienen der Veröffentlichung von Internet-Standards etc.
RIP	Routing Information Protocol – ein einfaches Distance-Vector-IGP
SAAC	Stateless Address Autoconfiguration – auf ND basierendes Verfahren zur automatischen Adresszuweisung
SNMP	Simple Network Management Protocol – Netzwerk-Management-Protokoll in IP-Netzen
SSH	Secure Shell
VLAN	Virtual LAN nach IEEE 802.1Q zum Einsatz in Ethernet-Netzwerken
VPN	Virtual Private Network – gesicherter Zusammenschluss mehrerer örtlich getrennter Netzwerke über das Internet
WAN	Wide/World Area Network – Netzwerk mit einer großen geographischen Ausdehnung

Glossar

Aggregation	Zusammenfassen mehrerer Routing-Tabellen-Einträge zu einem möglichst umfassenden Eintrag
Anycast	Versand einer Nachricht an einen Empfänger aus einer Gruppe – i. d. R. den topologisch nächstliegenden
Backbone	Der Teil eines Netzwerks, mit dem keine End-Systeme, sondern nur Router verbunden sind. Hier kommen i. d. R. die durchsatzstärksten Leitungen zum Einsatz.
Broadcast	Versand einer Nachricht an alle angeschlossenen Empfänger
Carrier	Leitungsnetzbetreiber – Besitzer der physischen Übertragungsstrecken eines WANs. Je nach Typ ist das Netz über eine Stadt (City-Carrier), ein Land (National Carrier) oder den Globus verteilt (Global Carrier).
Community	Passwort zum Zugriff auf einen SNMP-Agenten
Firewall	Ein Konzept zur Sicherung von Netzwerken, zu deren Umsetzung auch Paket-Filter eingesetzt werden.
Multicast	Versand einer Nachricht an alle Empfänger, die einen bestimmten Nachrichtendienst abonniert haben
OSI-Referenz-Modell	Modell zur Beschreibung von Netzwerk-Protokollen bestehend aus sieben Schichten. Diese dienen den folgenden Zwecken: Schicht 1: Bitübertragung, Schicht 2: Sicherung, Schicht 3: Vermittlung, Schicht 4: Transport, Schicht 5: Sitzung, Schicht 6: Darstellung, Schicht 7: Anwendung
Unicast	Versand einer Nachricht an einen exakt bestimmten Empfänger

1 Einführung und Motivation

Bereits vor dem Internet-Boom Ende der 90er Jahre des 20. Jahrhunderts wurde den für die Internet-Entwicklung verantwortlichen Gremien *IETF* (Internet Engineering Task Force) und *IESG* (Internet Engineering Steering Group) bewusst, dass das *Internet-Protokoll* (IP) künftigen Anforderungen nicht mehr gerecht werden wird. Fortschreitende Adress-Knappheit und das starke Wachstum der Routing-Tabellen wurden als dringlichste Probleme identifiziert und durch kurzfristige Maßnahmen wie *Network Address Translation* (NAT) und *Classless Interdomain Routing* (CIDR) entschärft. Man war sich aber bewusst, dass auf diese Weise der Kollaps nur hinausgezögert wird und adäquater Ersatz für IP geschaffen werden musste. Dieser Ersatz bekam den Namen *IPv6* (Internet Protokoll Version 6). Insbesondere in Asien und bisher unterentwickelten Regionen stellt IPv6 die einzige Lösung zur Überwindung des IPv4-Adress-Mangels dar.

Bei der Entwicklung der IPv6-Standards wurde zwar auf eine größtmögliche Kompatibilität mit dem alten Internet-Protokoll (IP Version 4 – kurz IPv4) gesetzt, aber einige entscheidende Veränderungen waren nötig, um den Anforderungen moderner Netzwerke gerecht zu werden.

Naturgemäß ist die Einführung eines neuen Standards als Ersatz für ein etabliertes Protokoll mit vielen Hürden verbunden. Es tritt in der Regel die Henne-Ei-Problematik auf: Die Hersteller von Hard- und Software entwickeln ihre Produkte vorrangig nach den Forderungen der Kunden, während viele Kunden nicht bereit sind, für neue, unausgereifte Produkte Geld auszugeben, wenn die alten Systeme problemlos arbeiten.

Dies hat dazu geführt, dass bis heute nur die Router weniger Hersteller eine für den praktischen Einsatz geeignete Unterstützung von IPv6 vorweisen können. Dagegen bieten die Hersteller von Betriebssystemen größtenteils seit mehreren Jahren eine stabile IPv6-Implementation an. Die freien Betriebssysteme Linux und die BSD-Derivate spielen bei dieser Entwicklung eine Vorreiter-Rolle und entsprechen dem aktuellen Stand der Standardisierung.

Die *Logivision GmbH*, mit deren Unterstützung diese Arbeit entstand, ist ein mittelständischer Internet-Service-Provider (ISP). Sie bietet einigen hundert Kunden unterschiedliche Internet-Dienstleitungen an und betreibt ca. 20 eigene Router. Für ein solches Unternehmen ist es von größter Bedeutung, innovative Dienste bei geringen Kosten anzubieten.

Da handelsübliche PCs aktuell über genügend Leistung verfügen, um die benötigten Bandbreiten von einigen Netzwerk-Interfaces im Bereich von 100 bis 155 $^{\text{MBit}}/_{\text{s}}$ oder sogar 1 $^{\text{GBit}}/_{\text{s}}$ zu bedienen, liegt es nahe, deren Einsatz als Router in Betracht zu ziehen. Dies bietet beim Einsatz von Open-Source-Software (OSS) eine große Flexibilität bezüglich innovativer Dienste, die von anderen Herstellern auf ihren proprietären Plattformen noch nicht implementiert wurden. Gleichzeitig sind PCs und deren Peripherie vergleichsweise sehr preiswert.

Auf Grund dieser Tatsachen soll diese Arbeit untersuchen, in wie weit der Einsatz eines PCs mit dem Betriebssystem Linux als Router den Ansprüchen eines Internet-Service-Providers gerecht wird und welche Schritte zur Realisierung nötig sind. In dieser Arbeit werden nur grundsätzliche Betrachtungen bezüglich der zu verwendenden Hardware angestellt. Zusätzlich soll diese Arbeit auch als Anleitung zum Aufbau eines solchen Routers dienen.

Für den sinnvollen Einsatz des Routers ist es allerdings nach wie vor wichtig, IPv4 zu unterstützen, da bisher nur wenige Kunden darauf verzichten können und IPv6 für diese höchstens ein Feature darstellt, das vielleicht in Zukunft interessant werden könnte.

Innerhalb dieser Arbeit werden die folgenden typografischen Konventionen verwendet:

- `Courier`: Es handelt sich um Kommandos, Quelltexte, Hostnamen, Dateien oder einen Verzeichnispfad.
- *Kursiv*: So werden Eigennamen von Produkten, Herstellern, Organisationen etc. gekennzeichnet.
- Das Zeichen „↵“ wird in Listings verwendet, um anzuzeigen, dass die Zeile noch nicht beendet ist, sondern der Rest in der nächsten Zeile folgt.

Diese Arbeit gliedert sich in die folgenden Themen-Bereiche:
Kapitel 2 beschreibt die technischen Grundlagen der IPv6-Protokoll-Familie und das Routing im Internet. Hier werden auch Technologien vorgestellt, die es ermöglichen die physischen Ports von Routern effizienter auszunutzen. Des Weiteren wird die grundlegende Funktion des Netzwerk-Managements erläutert.

In Kapitel 3 werden die Anforderungen an einen IPv6-Router analysiert. Es erfolgt auch eine Betrachtung der Schwierigkeiten, mit denen die Administratoren konfrontiert sind, wenn sie vor der Integration von IPv6 in ihrem Netzwerk stehen.

Anhand der Feststellungen der Analyse wird in Kapitel 4 ein Anforderungskatalog aufgestellt, nach dem der Router im darauf folgenden Kapitel aufgebaut wird. Dabei wird von einem Linux-Grundsystem ausgehend eine Anleitung zur Konfiguration der notwendigen Software gegeben. Schritt für Schritt wird der Router in das interne und das externe Routing des Provider-Netzwerks eingebunden und anschließend um zusätzliche Dienste ergänzt.

In Kapitel 6 wird eine Test-Umgebung vorgestellt, mit deren Hilfe der aufgebaute Router getestet werden kann. Die verschiedenen Tests werden dort beschrieben und mit deren Ergebnissen dargestellt.

Neben dem Vergleich mit einem Router des Herstellers *Cisco Systems, Inc.* werden in Kapitel 7 die Ergebnisse ausgewertet und die Tauglichkeit des Linux-basierten Routers für den Einsatz unter realen Bedingungen bewertet. Den Abschluss bildet ein Ausblick auf künftige Entwicklungsmöglichkeiten des IPv6-Routings.

2 Technische Grundlagen

Das Internet ist eine Ansammlung von Netzwerken, deren Gemeinsamkeit in der Nutzung der durch die *IETF* beschlossenen Standards besteht. Diese Standards werden in Form von *RFCs* (*Request for Comments*) veröffentlicht. Am Prozess der Standardisierung kann sich prinzipiell jeder durch seine Teilnahme an Diskussionen in den *Working Groups* beteiligen.

2.1 IPv6

2.1.1 Geschichte

Seit der bereits im Jahr 1973 stattgefundenen Entwicklung des Internet-Protokolls IPv4 wurden viele Erkenntnisse über paketvermittelnde Netzwerk-Technologien gewonnen. Die auf Grund dieser Einsichten angestrebten Entwicklungen konnten auf der Grundlage von IPv4 nur sehr schwer oder gar nicht umgesetzt werden. Deshalb fällten die Ingenieure der IETF bereits 1990 – weit vor der Entwicklung des World-Wide-Web (WWW) und dem dadurch angestoßenen Internet-Boom – die Entscheidung, IPv4 durch ein neues Protokoll zu ersetzen.

In der Folge stellten mehrere Entwickler-Teams ihre konkurierenden Vorschläge vor. Um diese Kräfte zu bündeln wurde 1993 das RFC1550 [BM93] mit der Aufforderung zu Vorschlägen für „IP: Next Generation“ veröffentlicht. Dieses Dokument schlug diskussionswürdige Themen vor, die zu einer genaueren Analyse der Anforderungen an das IPng in RFC1726 [PK94] führte. Daraufhin wurde laut RFC1752 [BM95] entschieden, das *Simple Internet Protocol Plus* (SIPP) in weiten Teilen zu übernehmen.

Da in jedem IP-Paket am Anfang eine Versions-Nummer untergebracht ist, musste auch IPng eine solche zugeteilt werden. Weil die Version 5 bereits für ein experimentelles Protokoll vergeben war, fiel die Entscheidung auf Version 6. Somit bekam das in RFC1883 und später RFC2460 [DH98] spezifizierte neue Protokoll den offiziellen Namen IPv6. Damit war bereits Ende 1995 der Weg für die technische Umsetzung formal frei.

Zu diesen Zeitpunkt begann mit dem *6bone* der Aufbau eines experimentellen, auf Tunneln basierenden IPv6-Internets. Auf diese Weise war es möglich, Erfahrungen mit dem neuen Protokoll zu sammeln, um sie später in Produktiv-Netze einfließen zu lassen. In den folgenden Jahren wurden verschiedene Hard- und Software-Produkte mit Unterstützung für IPv6 ausgestattet, weitere Analysen über die notwendigen Schritte zur Einführung des Protokolls veröffentlicht und die bei der Umsetzung erkannten Fehler durch neue Revisionen der Standards behoben. Vor allen Dingen aber mussten viele andere Protokolle an die neuen Erfordernisse angepasst werden.

2.1.2 Protokoll-Eigenschaften

Bei der Entwicklung von IPv6 wurden folgende Entscheidungen getroffen [DH98].

- Erweiterung des Adressraums von 32 Bit auf 128 Bit mit der Möglichkeit einer mehrstufigen Hierarchie und der Angabe von Gültigkeits-Bereichen um das Routing zu erleichtern.
- Beschleunigung der Paket-Verarbeitung durch eine Vereinfachung der Paket-Header.
- Verwendung von Erweiterungs-Headern für nicht ständig benutzte Optionen und Parameter die von transportierenden Systeme nicht ausgewertet werden brauchen.
- Erfüllung der Sicherheits-Bedürfnissen Authentizität und Schutz der Privatsphäre durch kryptografische Verfahren.
- Zur Erleichterung der Nutzung soll eine weitgehende Automatisierung der Konfiguration möglich sein.
- Die Umstellung von IPv4 auf das neue Protokoll muss schrittweise erfolgen und mit geringe Änderung in der übrigen Architektur verbunden sein.

2.1.3 IPv6-Adressen

Damit die Adressen, die vom neuen Internet-Protokoll verwendet werden, den Bedingungen bezüglich Hierarchisierung und automatischer Konfiguration genügen, musste der Adressraum sehr stark erweitert werden. Vor allen Dingen ist eine feste Aufteilung zwischen Netzwerk- und Host-Teil der Adresse definiert. Somit stehen von den 128 Bit der gesamten Adresse 64 Bit für die Adressierung des Netzwerks und 64 Bit für die Interface-ID (z. B. für das Netzwerk-Interface eines Computers) zur Verfügung [HD03]. Es ist vorgesehen, dass ein Computer mehrere IPv6-Adressen pro Interface besitzen kann.

Für IPv6 sind die folgenden Arten der Ziel-Adressen für bestimmte Zustellarten definiert:

Unicast : Zustellung zu einem bestimmten Interface eines Computers,

Anycast : Zustellung zu einem Ziel aus einer Gruppe von Interfaces, z. B. zum nächsten Nameserver,

Multicast: Zustellung zu allen Computern einer Gruppe.

Des Weiteren besitzen diese Adressen einen Gültigkeitsraum (*scope*), der die Weiterleitung wie folgt beschränkt:

link-local: Weiterleitung nur auf einem bestimmten physischen Netzwerk-Segment,

site-local: Weiterleitung nur innerhalb einer Organisation,

global : Weiterleitung im gesamten Internet möglich.

Da der Begriff *Site* unklar definiert ist und es beim Zusammenschluss unterschiedlicher Organisationen zu Problemen kommen kann, sollen die Site-Local-Adressen in ihrer bisher definierten Form abgeschafft werden [HC04]. Sie werden daher in dieser Arbeit nicht weiter beleuchtet.

Die einfache textliche Darstellung von IPv4-Adressen durch vier dezimale 8-Bit-Zahlen, die durch einen Punkt getrennt werden, ist für die Darstellung einer 128-Bit-Zahl wegen der Länge nur schlecht geeignet. Daher wurde entschieden, eine Schreibweise mit acht durch einen Doppelpunkt getrennten 16-Bit-Hexadezimal-Zahlen einzuführen. Mehrere zusammenhängende Gruppen von 16 Bit Nullen können zur Verkürzung einmalig durch zwei Doppelpunkte (`::`) ersetzt werden. Die in Abbildung 2.1 dargestellten IPv6-Adressen sind somit identisch.

```
1234:5678:9000:0000:0000:0000:00ab:cdef
1234:5678:9000:0:0:0:ab:cdef
1234:5678:9000::ab:cdef
```

Abbildung 2.1: Unterschiedliche Schreibweisen einer IPv6-Adresse

Zur vereinfachten Darstellung von IPv4-Adressen im IPv6-Adressraum, können die letzten 32 Bit einer Adresse auch durch die angestammte IPv4-Schreibweise ersetzt werden. So ergäbe sich aus der IPv4-Adresse `10.0.0.1` die kompatible Adresse `::10.0.0.1` bzw. `::ffff:10.0.0.1`.

Zur Identifikation eines Netzwerks dient eine an die Notation des Classless-Interdomain-Routings (CIDR) angelehnte Schreibweise des Präfixes. Dabei wird eine IPv6-Adresse von links beginnend geschrieben und durch einen Slash (`/`) von der Präfix-Länge abgetrennt. Diese gibt der Anzahl der für das Netz signifikanten Bits an. In Abbildung 2.2 werden korrekte Notationen für das 60-Bit-Präfix `123400000000567` gezeigt.

```
1234:0000:0000:5670:0000:0000:0000:0000/60
1234::5670:0:0:0:0/60
1234:0:0:5670::/60
```

Abbildung 2.2: Gültige Schreibweisen eines IPv6-Präfixes

Verschiedene Bereiche des IPv6-Adressraums sind für bestimmte Zwecke vorgesehen (siehe Tabelle 2.1). Alle anderen Adressen können als global gültige Unicast-Adressen verwendet werden.

IPv6-Adresse	**Zweck**	**Vgl. IPv4**
::/128	unspezifiziert	0.0.0.0/32
::1/128	Loopback	127.0.0.1/32
ff00::/8	Multicast	224.0.0.0/4
fe80::/10	Link-lokal	–

Tabelle 2.1: Spezielle IPv6-Adressen und ihre IPv4-Entsprechungen

Nach dem Start vergibt ein IPv6-fähiger Rechner an alle seine Netzwerk-Schnittstellen eine link-lokale Adresse, die sich aus dem Präfix `fe80::/64` und einer i. d. R. nach dem Standard IEEE EUI-64[1] gebildeten Interface-ID zusammensetzt. Daraufhin können einem Interface weitere Adressen hinzugefügt werden.

[1]Die 64 Bit lange EUI-64-Adresse wird im Falle von Ethernet aus der48 Bit umfassenden MAC-Adresse gewonnen.

2.1.4 Der IPv6-Header

Bei der Entwicklung des IPv6-Headers (siehe Abbildung 2.3) wurde darauf geachtet, dass nur solche Daten darin enthalten sind, die für eine effiziente Weiterleitung der Pakete notwendig sind. Zur Steigerung der Effizienz tragen die feste Größe von 40 Bytes und die Anordnung der Absender- und Empfänger-Adressen auf Offsets mit einem Vielfachen von 64 Bit bei. Dadurch können diese Informationen gut mittels spezieller Hardware ausgewertet werden. Die Bedeutung der einzelnen Felder wird in Tabelle 2.2 erläutert.

Damit das Protokoll auch in Zukunft erweiterbar ist, wurde ein System von optionalen Erweiterungs-Headern eingeführt. Informationen für Funktionalitäten wie Fragmentierung und zusätzliche Optionen werden in diese verlagert. Ein Vergleich mit dem IPv4-Header findet sich im Anhang B.

2.1.5 Die Extension-Header

Nicht immer reichen die Informationen des IPv6-Headers aus, um alle benötigten Informationen im Paket zu transportieren. Um diese unterzubringen, wurde für IPv6 ein System sog. *Extension-Header* entwickelt. Dieses basiert auf einer verketteten Liste. In jedem Teil-Header wird im Feld *Next Header* vermerkt, von welchem Typ der folgende Header sein wird. Abbildung 2.4 zeigt ein mögliches Beispiel.

Die Extension-Header werden i. d. R. nur von dem im IPv6-Header angegebenen Empfänger ausgewertet. Problematisch ist diese Tatsache im Zusammenhang mit Paket-Filtern (Firewall). Diese müssen zur Inspektion des Transport-Protokolls (z. B. TCP) die gesamte Kette der Erweiterungs-Header durchlaufen, was erhebliche Zeit in Anspruch nimmt. Außerdem ist für die Erweiterungen nicht zwingend eine Längenangabe vorgesehen, weshalb die Einführung eines neuen Typs dazu führt, dass Paket-Filter diesen noch nicht kennen und somit nicht bis zum Transport-Protokoll vordringen können.

Derzeit sind die folgenden Extension-Header definiert und sollen in dieser Reihenfolge angegeben und ausgewertet werden:

1. Hop-by-Hop Options: Optionen, die von den durchlaufenen Routern ausgewertet werden sollen.

2. Routing : Informationen zum Source-Routing und zur Aufzeichnung der von einem Paket durchlaufenen Router.

3. Destination Options : Optionen, die erst vom Ziel-System ausgewertet werden sollen.

4. Fragment : Ermöglicht die Fragmentierung eines Pakets beim Absender, das für die Übertragung über den vorgesehenen Weg zum Ziel zu groß ist.

Eine besondere Bedeutung kommt dem *Fragment Header* zu, da die Fragmentierung von Paketen bereits vom Absender durchgeführt wird und von den durchlaufenen Routern keine zusätzliche Veränderungen vorgenommen werden müssen. Zur Ermittlung der maximalen

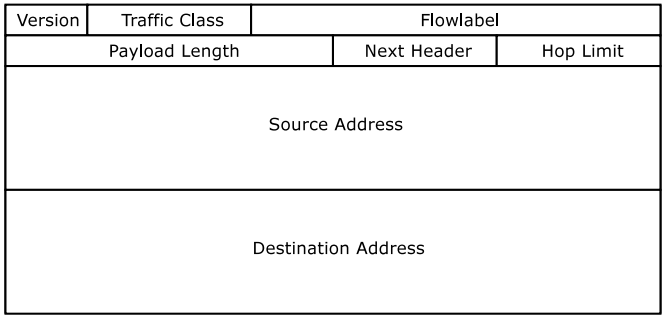

Abbildung 2.3: Schematischer Aufbau des IPv6-Header

Feld	Bedeutung
Version	Version des Internet Protokolls = 6
Traffic Class	Ermöglicht die Unterscheidung verschiedener Dringlichkeiten (derzeit experimentell)
Flowlabel	Ermöglicht effizientes Label-Switching in den passierenden Routern. Dazu ist eine Aushandlung des Flowlabels mit allen beteiligten Routern notwendig.
Payload Length	Anzahl von Bytes, die dem IPv6-Header in diesem Paket folgen
Next Header	Angabe des folgenden Headers – gibt entweder einen Erweiterungs-Header oder ein Layer-4-Protokoll an
Hop Limit	Wird von jedem durchlaufenen Router um eins vermindert. Bei Erreichen der Null wird das Paket verworfen.
Source Address	Adresse des Absenders
Destination Address	Adresse des Empfängers – bei Vorhandensein eines Routing-Headers kann dies die Adresse einer Zwischenstation sein

Tabelle 2.2: Bedeutung der Felder im IPv6-Header

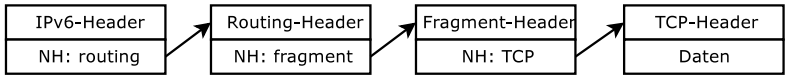

Abbildung 2.4: Beispiel eines IPv6-Pakets, das Extension-Header beinhaltet

Paketgröße für die Zustellung zum Ziel (*Path MTU*) dient die *Path MTU Discovery*. Dabei sendet eine Zwischenstation, die das Paket auf Grund seiner Größe nicht weiter transportieren kann, eine Fehlermeldung per ICMP (*Internet Control Message Protocol*) mit der maximal möglichen MTU. Mit Hilfe dieser Information wird das Paket entsprechend fragmentiert erneut gesendet. Zusätzlich sollte sich der Absender die Path MTU merken, um künftige Pakete gleich mit einer korrekten Fragmentierung abzusenden. Die MTU darf laut IPv6-Standards aber nie kleiner als 1280 Byte sein, so dass eine hinreichend effiziente Übertragung gewährleistet ist.

2.1.6 Automatische Konfiguration

Um einen möglichst einfachen Einsatz von IPv6 zu ermöglichen, sollen sich die beteiligten Systeme zu einem großen Teil automatisch konfigurieren können. Zu diesem Zweck wurden Erweiterungen auf Basis des ICMPv6 (*ICMP für IPv6*) eingeführt und in RFC2461 [NNS98] sowie RFC2462 [TN98] definiert.

Die darin vorgestellten Verfahren umfassen die folgenden Funktionen:

- *Duplicate Address Detection* (DAD) zur Erkennung der mehrfachen Verwendung einer Adresse,
- *Neighbor Discovery* (ND) zur Übersetzung der IPv6-Adresse in eine Adresse der darunterliegenden Schicht des OSI-RM,
- *Router Discovery* (RD) zum Auffinden der zuständigen *Default-Gateways* und zur Übermittlung des zu verwendenden, globalen Adress-Präfixes.

Wird ein Host mit IPv6-Unterstützung an ein Netzwerk angeschlossen, gibt er sich zuerst eine link-lokale Adresse für das betreffende Netzwerk-Interface. Daraufhin sendet er per Multicast eine Nachricht an alle an dieses Netzwerk angeschlossenen Router mit der Aufforderung sich zu melden (*Router Solicitation*). Alle vorhandenen Router antworten daraufhin mit einem *Router Advertisement*, das neben ihrer Adresse auch weitere Parameter, wie die zu verwendende MTU und die vorgesehenen globalen Adress-Präfixe, enthält. Daraufhin bildet der Host aus den ersten 64 Bit des Präfixes und den letzten 64 Bit seiner link-lokalen Adresse eine neue Adresse. Nach einer Prüfung, die sicherstellen soll, dass diese Adresse von keinem anderen System benutzt wird, wird sie an das Interface gebunden. Dieses gesamte Verfahren wird *Stateless Address Autoconfiguration* (SAAC) genannt.

Neben dieser Methode, bei der keine Informationen über die bereits vergebenen Adressen zentral gespeichert werden, wird auch eine Konfiguration mittels DHCPv6 (*Dynamic Host Configuration Protocol für IPv6* [BVL+03]) angeboten. Auf diese Weise lässt sich die Vergabe der Adressen zentral steuern. Ein weiterer Vorteil besteht in der Fähigkeit zusätzliche Informationen zu übermitteln. Dazu gehören z. B. die Adressen der *Domain-Name-Server* (DNS) [Dro03]. Für den Anschluss von Kunden-Netzwerken an einen ISP muss dem Kunden-Router das zu verwendende Adress-Präfix[2] mitgeteilt werden. Auch diese Aufgabe

[2] Laut Adressvergabe-Richtlinie wird einem Kunden i. d. R. ein 48-Bit-Präfix zugeteilt.

erfüllt DHCPv6 mittels *Prefix Delegation* [TD03]. Auf diese Weise ist ein hierarchisches System von DHCPv6-Servern denkbar, mit dem Adress-Bereiche verteilt werden.

Somit lassen sich im Idealfall sämtliche zur Verwendung von IPv6 notwendigen Konfigurationen in einem angeschlossenen Netzwerk automatisieren. Da sich wichtige Teile der Infrastruktur (z. B. Router und Server) nicht auf die korrekte Funktion anderer Systeme verlassen sollte, wird bei diesen Geräten in der Praxis eine manuelle Konfiguration der Adressen vorgenommen.

2.1.7 IPSec

Da in der Vergangenheit mehrfach Angriffe auf IP-Infrastruktur durchgeführt wurden, definiert IPv6 die Unterstützung von IPSec zur Absicherung der Übertragung gegen Veränderung bzw. Mitlesen als Pflicht. Die Integration dieser Funktionen erfolgt ebenfalls in Form von Extension-Headern.

IPSec wird für einen IPv6-Router immer dann benötigt, wenn Routing-Updates über den selben Datenkanal wie die Nutzdaten übertragen werden, was dem Regelfall entspricht. Viele Protokoll-Spezifikationen, die auf IPv6 basieren, geben zur Absicherung gegen Angreifer IPSec als die einzige Möglichkeit an. Daher kommt der Unterstützung und dem Einsatz von IPSec eine besonders große Bedeutung zu.

Die Spezifikationen für IPSec sind sehr umfangreich und sehen verschiedene Betriebsmodi vor [AGL03]. Neben einem Transport-Modus, bei dem nur die Nutzdaten abgesichert werden, existiert ein Tunnel-Modus, mit dem komplette IP-Pakete gesichert transportiert werden können. Daneben gibt es zwei verschiedene Arten der Absicherung, den *Authentication Header* (AH) und den *Encapsulating Security Payload* (ESP). Während ersterer nur der Integritäts-Sicherung durch einen kryptografischen Hash (z. B. MD5, SHA1) dient, ermöglicht letzterer auch die Verschlüsselung mittels symmetrischer Algorithmen (z. B. 3DES, AES).

2.2 Internet-Routing

Die Datenpakete der Internet-Protokolle IPv4 und IPv6 beinhalten in ihren Paketköpfen u. a. Informationen über den Absender und Empfänger der transportierten Nachricht. Im Normalfall entscheiden Router auf Grund der Empfänger-Adresse, wohin sie das entsprechende Paket weiterleiten. Bis der Empfänger erreicht wird, kann eine Nachricht auf diese Weise also mehrere Zwischenstationen passieren.

Um diese Weiterleitungs-Entscheidung treffen zu können, benötigen die Router Informationen darüber, auf welchem Weg das Ziel zu erreichen ist. Eine Möglichkeit besteht darin, jedem Router die Routing-Informationen manuell in seiner Routing-Tabelle zu konfigurieren. Dieses *statische Routing* hat den Nachteil, dass es sich an veränderliche Bedingungen nicht automatisch anpassen kann. Um die Änderungen der Topologie komplexer Netzwerke durch Leitungs-Ausfälle etc. im Routing berücksichtigen zu können, bedarf es daher eines *dynamischen Routings*.

Dazu müssen sich die Router über die von ihnen erreichbaren IP-Netze austauschen. Die dafür notwendigen Routing-Protokolle erfüllen unterschiedliche Anforderungen und sind daher unterschiedlich komplex.

Allgemein werden Routing-Protokolle danach unterschieden, ob sie innerhalb eines Netzes mit einheitlicher Administration (*Domain*) eingesetzt werden oder zwischen solchen Domains.

2.2.1 Intra-Domain-Routing

Zwischen den Routern einer Routing-Domain, z. B. dem Netzwerk eines ISP, werden Interior-Gateway-Protokolle (IGP) benutzt, um die Routing-Informationen auszutauschen. Dabei wird zwischen Distance-Vector- und Link-State-Protokollen unterschieden, von denen hier jeweils ein wichtiger Vertreter kurz vorgestellt wird. Ausführlichere Informationen dazu finden sich z. B. in [Hui00].

2.2.1.1 RIP

Eines der ältesten und einfachsten Routing-Protokolle für IP-Netzwerke ist das *Routing Information Protocol* (RIP). Von ihm gibt es auf Grund geänderter Anforderungen mehrere Versionen, unter anderem auch für IPv6. Auf Grund des ursprünglichen Namens IPng wurde es RIPng genannt.

Die Einfachheit rührt vom zu Grunde liegenden Distance-Vector-Prinzip her. Dabei sendet jeder Router die ihm bekannten Routing-Informationen aus Ziel und einer zugehörigen Metrik (erhöht um eins oder einen anderen festen Wert) in kurzen Abständen an alle seine Nachbarn. Beim Empfang einer solchen Information fügt ein Router die Route seiner Routing-Tabelle hinzu, wenn sie bisher nicht bekannt war oder die Metrik kleiner als der bekannte Wert ist. Wenn ein Router in einem bestimmten Zeit-Intervall kein Update empfängt, lässt er die betreffenden Einträge altern (Erhöhung der Metrik um eins). Mit Erreichen einer maximalen Metrik (bei RIP 16) wird ein so „überalterter“ Eintrag aus der Tabelle entfernt. Bis dahin gilt das Ziel aber noch als erreichbar, was zu Routing-Schleifen führen kann.

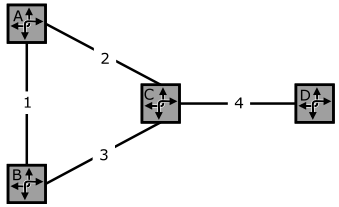

Abbildung 2.5: IGP-Beispiel-Netzwerk mit vier Routern

Anhand eines Beispiels mit vier Routern (siehe Abbildung 2.5) soll nun der Prozess vom Start bis zum Konvergieren des Routings demonstriert werden. Zur Vereinfachung stehen die Router A bis D jeweils stellvertretend für die an sie angeschlossenen Netzwerke.

Nach dem Start jedes Routers kennt dieser nur die direkt an ihn angeschlossenen Netzwerke. Diese Information sendet er per Multicast mit einer Metrik von 1 an seine Nachbarn:

A →	Link	Wert	B →	Link	Wert	C →	Link	Wert	D →	Link	Wert
A	**lokal**	**0**	**B**	**lokal**	**0**	**C**	**lokal**	**0**	**D**	**lokal**	**0**

Daraufhin lernen die benachbarten Router die zusätzliche Einträge und senden sie mit einer erhöhten Metrik an alle anderen Nachbarn.

A →	Link	Wert	B →	Link	Wert	C →	Link	Wert	D →	Link	Wert
A	lokal	0	B	lokal	0	C	lokal	0	D	lokal	0
B	**1**	**1**	**A**	**1**	**1**	**A**	**2**	**1**	**C**	**4**	**1**
C	**2**	**1**	**C**	**3**	**1**	**B**	**3**	**1**			
						D	**4**	**1**			

Zwar empfängt Router C auch ein Update von Router B mit dem Ziel A und einer Metrik von 2, ignoriert dieses allerdings, da eine kürzere Route bereits in der Tabelle enthalten ist.

A →	Link	Wert	B →	Link	Wert	C →	Link	Wert	D →	Link	Wert
A	lokal	0	B	lokal	0	C	lokal	0	D	lokal	0
B	1	1	A	1	1	A	2	1	C	4	1
C	2	1	C	3	1	B	3	1	**A**	**4**	**2**
D	**2**	**2**	**D**	**3**	**2**	D	4	1	**B**	**4**	**2**

Nun hat das Routing den angestrebten stabilen Zustand erreicht, es ist konvergiert.

Wenn sich die Topologie verändert, indem z. B. eine Leitung nicht mehr verfügbar ist und somit keine Updates mehr über diese ankommen, muss sich das Routing anpassen.

Fällt z. B. Link 2 aus, erhöht Router C seinen Eintrag in Richtung A auf den maximalen Wert, was *unerreichbar* bedeutet. Erst nach einem erneuten Update von Router B lernt C wieder einen neuen Weg zu A mit einer Metrik von 2 kennen.

Für komplexe Netze kann dies bedeuten, dass das Routing nur langsam konvergiert und einige Ziele zeitweise unerreichbar sind. Daher sollten RIP und andere Distance-Vector-Protokolle auch in kleinen Netzwerken nur eingesetzen werden, wenn es keine Alternative gibt.

2.2.1.2 OSPF

Ein Vertreter der Link-State-Protokolle ist das von der IETF entwickelte *Open Shortest Path First*. Es implementiert den *Shortest-Path-First*-Algorithmus von E. W. Dijkstra als offenen Standard. Von diesem ursprünglich für IPv4 entwickelten Protokoll existiert auch eine in RFC2740 [CFM99] definierte Version für IPv6 – OSPFv3.

Link-State-Protokolle zeichnen sich dadurch aus, dass jeder Router eine komplette Karte des Netzwerks kennt. Dies erfordert, dass jeder einzelne Router aus diesem Graphen einen Baum mit allen erreichbaren Zielen errechnet. Um Schleifen im Routing zu vermeiden, müssen alle beteiligten Router den selben Algorithmus verwenden. Dieser Vorgang kostet natürlich Speicher und Rechenzeit und ist somit aufwändiger als das Vorgehen bei RIP. Zusätzlich ermöglicht OSPF eine größere Auswahl von Metriken, als die Zahl der durchlaufenen Router. Pro durchlaufenem Link wird ein Kosten-Wert definiert. Je geringer die Kosten, desto eher wird ein entsprechender Link verwendet. Somit lassen sich wesentlich feinere Routing-Entscheidungen und die Nutzung von mehreren Wegen zu einem Ziel (*Equal Cost Multipath Routing*) herbeiführen.

Der größte Vorteil dieser Protokollfamilie liegt in der schnellen Konvergenz, da nach jeder Änderung der Netzwerk-Topologie alle Router davon in Kenntnis gesetzt werden. Dieses Fluten (*Flooding*) wird erreicht, indem jeder Router die neuen Informationen unverzüglich an alle anderen Nachbarn weitersendet. Auf diese Weise wird die grundlegende Link-State-Datenbank aller Router abgeglichen.

Neben der Übertragung der Updates durch Fluten muss OSPF auch andere Kommunikationen durchführen. Um die Verfügbarkeit einer Verbindung zu prüfen, werden in kurzen Intervallen *Hello*-Pakete an die Nachbarn versandt, die auch die Router-IDs der eigenen Nachbarn enthalten. Dadurch können auch Links mit unidirektionalem Ausfall erkannt werden.

Zusätzlich existiert das *Exchange-Protokoll*, mit dem einem neu aktivierten Router die gesamte Link-State-Datenbank übermittelt wird.

All diese unterschiedlichen Kommunikationen und Anforderungen machen OSPF sehr komplex, aber auch robust und vielseitig. Um die OSPF-Kommunikation abzusichern, wurde in OSPFv2 (OSPF für IPv4) eine Header-Erweiterung eingeführt, die die Prüfung der Authentizität per MD5-Hash ermöglicht. Dieses Vorgehen ist bei OSPFv3 durch das Vorhandensein von IPSec nicht mehr nötig, was das Protokoll etwas vereinfacht.

Für das Beispiel aus Abbildung 2.5 ergäbe sich nach dem vollständigen Austausch aller Informationen zur Erreichbarkeit die in Abbildung 2.6 dargestellte Link-State-Tabelle. Diese Abbildung zeigt auch die Baum-Struktur aus Sicht des Routers A. Dabei sollen beispielhaft für den Link 2 die Kosten verdoppelt werden, was dazu führt, dass die Router C und D über zwei gleichwertige Wege erreichbar sind.

von	nach	Link	Kosten
A	B	1	1
A	C	2	2
B	A	1	1
B	C	3	1
C	A	2	2
C	B	3	1
C	D	4	1
D	C	4	1

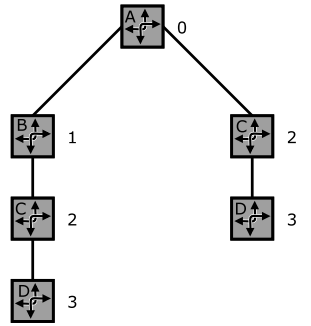

Abbildung 2.6: Link-State-Tabelle und Baum-Struktur aus Sicht von Router A

2.2.2 Inter-Domain-Routing

Beim Routing innerhalb eines Netzwerks, das einer einzigen Organisation gehört, sind nur wenige Metriken zu beachten. Diese umfassen meist nur die Anzahl der Zwischenschritte und die Bandbreiten der zwischen ihnen vorhandenen Leitungen. Beim Übergang in andere Netzwerke müssen dagegen vermehrt Entscheidungen über Kosten und weiche Faktoren (z. B. Vertrauen in einen Carrier) getroffen werden. Für diesen Zweck wurden *Exterior-Gateway-Protokolle* (EGP) entwickelt, denen lokale Entscheidungen als Parameter übergeben werden können.

Neben diesen Anforderungen ist es den IGPs auch nicht möglich, die große Zahl einzelner Routen in der DFZ[3] zu bewältigen. Auf dem in Kapitel 6 erwähnten Router, dem eine vollständige BGP-Routing-Tabelle zur Verfühung stand, wurden ca. 140.000 IPv4-Routen und ca. 500 IPv6-Routen registriert.

Im Inter-Domain-Routing werden die einzelnen zusammenzuführenden Netze auch *Autonomes System*[4] (AS) genannt.

BGP

Das Feld der EGPs beherrscht heute weitestgehend das *Border Gateway Protocol* (BGP) in der Version 4 [RL95]. Dies ist die erste BGP-Version gewesen, die CIDR für IPv4 unterstützt. Auf Grund der Entwicklung von Multi-Protokoll-Erweiterungen [BKR00] (oft BGP-4+ genannt) ist es auch in der Lage seine Funktion für IPv6 zu übernehmen.

Die vielen Routen, die in der DFZ ausgetauscht werden, machen die Nutzung eines Link-State-Protokolls wie OSPF unpraktikabel. Da auch das „Zählen bis Unendlich“, bis zum Erreichen einer maximalen Metrik und dem darauf folgenden Löschen einer Route, von Distance-Vector-Protokollen in einem zentralen Teil der Routing-Architektur nicht sinn-

[3] Default free zone – Gruppe der Internet-Router, die keine Default-Route besitzen und nur anhand der Daten aus dem EGP arbeiten

[4] von engl. Autonomous System

voll umsetzbar ist, musste für BGP ein anderes Verfahren gefunden werden. Dieses basiert auf einer TCP-Verbindung zwischen benachbarten Routern, deren Beendigung als Abbruch der physischen Verbindung gewertet wird. Zur Vermeidung von Routing-Schleifen wird zu jedem Ziel-Präfix der zu durchlaufene Weg durch die autonomen Systeme im `AS_PATH` mitgeführt. Teilweise wird das von BGP genutzte Prinzip deshalb auch als „Path-Vector-Verfahren“ bezeichnet.

Neben der Vermeidung von Schleifen dient der `AS_PATH` als wichtigste Merkmal für die Routing-Entscheidung, da die Zahl der durchlaufenen AS eine ungefähre Länge des benötigten Weges angibt. Dem `AS_PATH` wird beim Aussenden eines Updates an einen Router eines anderen AS die lokale AS-Nummer vorangestellt. Wenn Router A aus dem Beispiel in Abbildung 2.7 das Präfix `2001:db8:2000::/35` seinen Nachbarn als erreichbar anbieten möchte, gibt er ihm den `AS_PATH 65001` mit. Wenn Router D diese Information an Router C weiterleitet, enthält sie den `AS_PATH 65003 65001`.

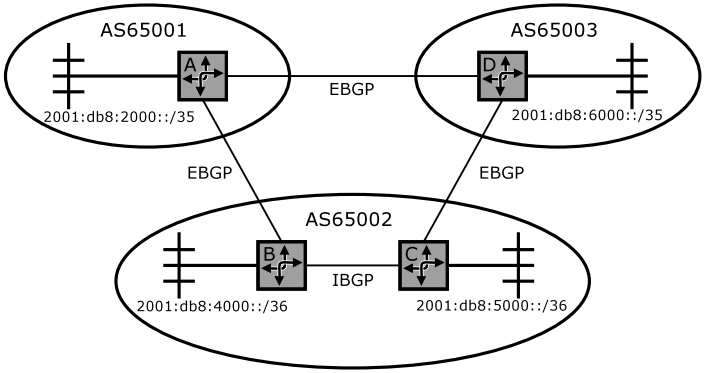

Abbildung 2.7: BGP Beispiel-Netzwerk mit drei AS

Das Beispiel zeigt auch den üblichen Fall, dass in einem AS mehrere Router mit Außenverbindung vorhanden sind. Um eine korrekte Funktion zu gewährleisten, müssen diese Router untereinander ebenfalls BGP-Verbindungen aufbauen. Wenn keine speziellen Techniken, wie *Route-Reflektoren* oder *Confederations*, zum Einsatz kommen, muss ein vollständig vermaschtes Netz der BGP-sprechenden Router eines AS aufgebaut werden. Ansonsten können Routing-Informationen fehlen. Da für diese internen Verbindungen andere Regeln gelten als für externe, wird zwischen iBGP (*Interior BGP*) und eBGP (*Exterior BGP*) unterschieden.

Bei einer iBGP-Verbindung wird der `AS_PATH` nicht modifiziert, sondern die Erreichbarkeits-Information unverändert weitergeleitet. Um eine korrekte Funktion von iBGP zu gewährleisten, sollte ein IGP eingesetzt werden. Da die IGPs schnellere Umschaltzeiten bieten, wird die durch das eigene Netzwerk bedingte Nicht-Erreichbarkeit auf ein Mindestmaß reduziert und die TCP-Verbindung problemlos aufrecht erhalten. Zusätzlich ergeben

sich dadurch für BGP weniger Topologie-Veränderungen, so dass weniger Update-Pakete an die BGP-Partner versandt werden müssen.

Um den Umfang der globalen Routing-Tabelle zu minimieren, sollten die anzubietenden Adress-Bereiche möglichst groß gehalten werden. Die Router aus AS65002 sollten ihren Nachbarn lediglich das Präfix `2001:db8:4000::/35` anbieten und die weitere interne Verteilung dem IGP überlassen. Außerdem bietet BGP die Möglichkeit, mehrere Präfixe zusammenzufassen. Hat das AS65002 aus dem obigen Beispiel etwa als Einziges eine Verbindung zum Internet und fungiert es somit als *Upstream-Provider* für die beiden anderen AS, können die drei Routen z. B. zu `2001:db8::/32` zusammengefasst werden. Um die Informationen aus dem `AS_PATH` nicht zu verlieren, werden alle AS in einem `AS_SET` so zusammengefasst, dass der `AS_PATH [65001 65002 65003]` an die Partner übermittelt wird.

Da der `AS_PATH` als alleiniges Merkmal für eine effiziente Routing-Entscheidung nicht ausreicht, werden auch andere Attribute verwendet, die mit den Routing-Updates übertragen werden oder lokal definiert sind. Die lokal definierten Bedingungen besitzen die höchste Priorität. Diese umfassen eine Gewichtung, die pro Router konfiguriert wird sowie das Attribut `local_pref`, das die *Routing-Policy* eines AS wiederspiegelt. Als weitere Kriterien, die mit den Updates verteilt werden, kommen das `ORIGIN`-Feld und die `MED`-Option zum Einsatz. In den *Multi-Exit Discriminator* (MED) kann die Metrik des IGPs eingehen, so dass bei mehreren Übergängen zwischen zwei AS der jeweils netz-topologisch günstigere gewählt werden kann.

Viele weitere Besonderheiten und Konfigurations-Beispiele können in [HM01] nachgelesen werden.

2.3 Virtuelle Netzwerk-Schnittstellen

Die physischen Netzwerk-Schnittstellen eines Routers sind eine begrenzte Ressource, deren Anschaffung viel Geld kostet. Daher wurden in den letzten Jahren viele Technologien entwickelt, um eine größere Zahl virtueller Verbindungen zur Verfügung stellen zu können. Einige Netzwerk-Technologien wie ATM (*Asynchronous Transfer Mode*) definieren auf Grund ihres Ursprungs in leitungsvermittelnden Netzen virtuelle Kanäle. Erst in den letzten Jahren wurden solche Fähigkeiten auch für das preiswerte Ethernet entwickelt.

Auch andere Erfordernisse, wie *Backbone-Performance* und die administrative Trennung von physischer und logischer Netzwerkanbindung, bedingen den Einsatz von virtuellen Verbindungen und Tunnel-Technologien.

2.3.1 VLAN

Mit der Umstellung auf eine sternförmige physische Struktur beim preiswerten Ethernet wurde der Einsatz von Multiport-Bridges (im Folgenden *Switches* genannt) immer interessanter. Durch voranschreitende Miniaturisierung in der gesamten IT wurden auch immer höher integrierte Switches mit mehreren hundert Ports verfügbar und erschwinglich. Gleichzeitig ist aber der Bedarf an logisch getrennten Netzwerken aus Sicherheitsgründen vorhanden. Um dies zu erreichen, können in diesen Geräten virtuelle LANs (VLANs) definiert werden. Dazu werden jeweils mehrere Ports in einem VLAN zusammengefasst. Der Bedarf einer effizienten hierarchischen Netzwerk-Verkabelung erfordert zudem die Möglichkeit, auf einer physischen Verbindung zwischen Switches mehrere VLANs weiterzuleiten.

Dazu wird den Ethernet-Frames nach IEEE 802.1Q ein 4 Byte langer VLAN-Tag vorangestellt, der eine Unterscheidung von bis zu 4096 VLANs ermöglicht [AG02].

Dieses Verfahren ermöglicht es nicht nur, Switches untereinander zu verbinden, sondern bietet auch die Möglichkeit, Router mit mehreren VLANs kommunizieren zu lassen. Die weitere Vermittlung übernimmt dann ein Switch, dessen physische Ports wesentlich preiswerter sind.

Abbildung 2.8 zeigt die physischen und die logischen Verbindungen dreier Router einmal ohne und einmal mit dem Einsatz von VLANs. In diesem Fall wird pro Router ein physisches Interface eingespart. Befinden sich noch mehr Router an einem Ethernet-Segment, wirkt sich dieses Verfahren u. U. noch wesentlich stärker aus. Ein solcher Fall tritt z. B. an Internet-Austauschpunkten auf, an denen dadurch virtuelle dedizierte Verbindungen zu mehreren Routern anderer Provider hergestellt werden können.

2.3.2 MPLS

Wenn ein Paket in das Netz eines Providers eintritt, bestimmt der erste Router (auch *Border-Router* genannt) den Router, der den besten Ausgang aus dem Netzwerk zur Verfügung stellt, und leitet das Paket dorthin weiter.

Bei herkömmlichem Routing wird ein IP-Paket von jedem passierten Router auf seine Ziel-IP-Adresse hin untersucht, die dafür beste Route in der Routing-Tabelle nachgeschlagen und an das entsprechende Interface weitergeleitet. Um diesen Vorgang zu beschleuni-

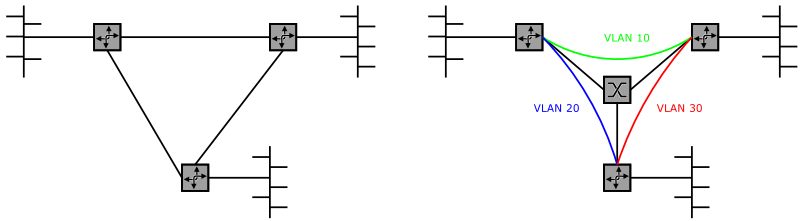

Abbildung 2.8: Verbindung von Routern ohne und mit VLAN

gen, bietet es sich an, die ersten beiden Schritte nur einmal durchzuführen. Dazu muss der erste Router dem IP-Paket allerdings eine Information über den zu verwendenden Weg mitgeben, die auf effiziente Weise von den Backbone-Routern in Hardware ausgewertet werden kann.

Da dies dem Switching auf Schicht 2 des OSI-RM sehr ähnlich ist, wurde dieses in RFC3031 ff. [RVC01] definierte Verfahren auch *Multiprotocol Label Switching* (MPLS) genannt. Dabei wird vor dem Schicht-3-Header ein Zwischen-Header eingefügt, der ein vereinbartes Label trägt, anhand dessen der Empfänger über die weitere Behandlung entscheiden kann.

Auch wenn der IPv6-Header bereits ein Feld namens *Flowlabel* anbietet, das den selben Zweck erfüllen kann, wurde eine solche Entwicklung für IPv4 nötig. Sie bietet u. a. die Möglichkeit, ohne Aufrüstung des Backbones um IPv6-Funktionalität IPv6-Pakete über Backbone zu transferieren.

Durch solche Flowlabel ergeben sich noch viele andere Vorteile. So lassen sich auf diese Weise für bestimmte Anwendungen verschiedene Wege definieren, um *Quality of Service* garantieren zu können. Weiterhin ist es möglich weit voneinander entfernte Standorte zu einem *Virtual Private Network* (VPN) zusammenzuschalten, wenn das dazwischen liegende Transportnetz in der Hand einer einzelnen Administration liegt. Dies ermöglicht es z. B. Internet-Service-Providern, auf kostengünstige Weise an mehreren Standorten aktiv zu sein, ohne dafür dedizierte Leitungen mieten zu müssen.

Diese Möglichkeiten ähneln denen von ATM, ermöglichen aber auch den transparenten Einsatz anderer Netzwerk-Technologien, die u. U. günstiger einzusetzen sind. Eine ausführlichere Betrachtung findet diese Technologie in [vL01].

2.3.3 L2TP

Zur Anbindung von Kunden an das Netz eines ISPs wird grundsätzlich das *Point-to-Point Protocol* (PPP) benutzt. Dies erfordert normalerweise eine direkte Verbindung zwischen dem Endkunden-System und einem Einwahl-Router. In einigen Fällen ist dies preislich ungünstig oder im Falle von ADSL, auf Grund beschränkter Entfernung, sogar unmöglich.

Ein Lösungsansatz besteht darin, die Behandlung der PPP-Session in zwei Teile zu trennen. Die eine Seite kümmert sich um die physische Terminierung der Leitung und Weiterleitung des PPP-Datenstroms an ein entferntes System, das auf der anderen Seite die

logische PPP-Session terminiert. Dieser Ansatz wird vom *Layer Two Tunneling Protocol* (L2TP) [PPR+99] implementiert.

Abbildung 2.9 zeigt den schematischen Aufbau einer solchen Verbindung. Der *L2TP Access Concentrator* (LAC) empfängt die PPP-Daten von der physischen Leitung und leitet diese an den *L2TP Network Server* (LNS) weiter, der im nachfolgenden Beispiel die Verbindung zum Internet herstellt. Neben den einzelnen L2TP-Sessions wird zwischen LAC und LNS auch ein Steuer-Kanal aufgebaut, über den die Verbindungs-Steuerung für die einzelnen Sessions stattfindet. L2TP basiert auf UDP zum Transport der Daten.

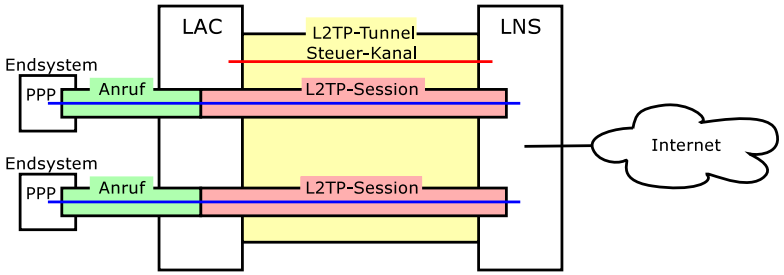

Abbildung 2.9: Schematischer Aufbau eines L2TP-Tunnels

Diese Technik ermöglicht es auch kleinen ISPs, die Einwahl-Technik größerer Carrier mitzunutzen und somit eine größere geographische Präsenz zu erreichen. Auf diese Weise werden z. B. in Deutschland von vielen ISPs Angebote auf Basis der ADSL-Anschlüsse der Firma *T-Com* gemacht. Des Weiteren ist der Einsatz von L2TP auch sinnvoll, wenn die Einwahl-Router bestimmte Funktionen des Backbones (IPv6, MPLS) nicht unterstützen oder nicht kostengünstig aufgerüstet werden können.

Eine andere häufig genutzte Möglichkeit zum Einsatz von L2TP ist das Zusammenlegen von LAC und Endsystem. Dadurch lässt sich PPP über das Internet übertragen und stellt somit eine flexible Möglichkeit zum Transport von verschiedenen Layer-3-Protokollen (z. B. IP, IPv6, IPX) über weite Strecken dar. Diese Art VPNs aufzubauen ist für einen ISP i. d. R. wenig bedeutsam.

2.4 Netzwerk-Management

Die Verwaltung einer großen Zahl von Netzwerk-Geräten, wie Switches, Router und Server, erfordert häufig viel Zeit und bindet somit wertvolle Arbeitskraft. Sind diese Geräte über verschiedene Standorte verteilt, müssen sie fernwartbar sein. Dazu steht meist eine fernbedienbare Konsole[5] per Telnet oder SSH[6] zur Verfügung.

Des Weiteren werden Netzwerk-Management-Systeme angeboten, die alle vorhandenen Endgeräte unter einer grafischen Oberfläche zusammenfassen. Dazu bedarf es einer einheitlichen Schnittstelle zwischen dem Netzelement und dem Management-System. Im Bereich der IP-Infrastruktur hat sich dazu das *Simple Network Management Protocol* (SNMP) durchgesetzt, das im Folgenden näher betrachtet werden soll.

SNMP

Das *Simple Network Management Protocol* [SNM02] bietet die Möglichkeit, Verwaltungs-Operationen an einem entfernten System vorzunehmen. Dazu gehört einerseits die Abfrage des aktuellen Status, andererseits auch die Änderung bestimmter Parameter. Dazu wird von einem SNMP-Manager (i. d. R. ein Netzwerk-Management-System) eine entsprechende Anforderung per UDP an den SNMP-Agent des Zielsystems gesendet und von diesem beantwortet. Um ein wichtiges Ereignis auch ohne Abfrage des Managers behandeln zu können, besteht die Möglichkeit, dass der Agent SNMP-Traps versendet.

Da derzeit kein frei verfügbares Netzwerk-Management-System existiert, werden in kleineren Netzwerken häufig nur Grundfunktionen genutzt. Dazu kommen meist Kommandozeilen-Programme und spezialisierte Skripte zum Einsatz.

Die verschiedenen per SNMP zur Verfügung gestellten Variablen werden über ein hierarchisches System von Objekt-IDs (OID) adressiert. Diesen OIDs liegt die *Management Information Base* (MIB) zu Grunde. Darin werden die verfügbaren Datentypen und der Baum der Variablen mit zugehörigen Namen definiert. Zur leichteren Lesbarkeit kann statt der numerischen OID auch eine textliche Repräsentation genutzt werden. Der namentliche Zugriff auf die Variable `iso.org.dod.internet.mgmt.mib-2.system.sysName` kann ebenso durch die etwas kryptische Darstellung `.1.3.6.1.2.1.1.5` ersetzt werden.

Neben der Abfrage eines einzelnen Wertes mittels eines *GetRequest*s, kann mit Hilfe von mehreren *GetNextRequest*s auch ein Teil des Baums durchlaufen werden, um mehrere Parameter abzufragen. Dieses Vorgehen ist auch notwendig, wenn eine Tabelle mit Werten zur Verfügung gestellt wird, deren Indizes vorher nicht bekannt sind.

Der Standard definiert sowohl das Format der übertragenen Nachrichten, als auch verschiedene Mechanismen des Zugriffs-Schutzes. Die Versionen 2 und 3 von SNMP bieten diesbezüglich eine sehr fein einstellbare Zugriffs-Kontrolle. Trotzdem setzen sich diese Versionen nur langsam durch. Stattdessen wird vor allen Dingen SNMPv2c eingesetzt, welches die einfache Zugriffs-Steuerung mittels *Communities* aus SNMPv1 übernommen hat. Eine Community stellt dabei ein einfaches Passwort zum Zugriff auf einen Agenten dar.

[5] auch CLI – *Command Line Interface*

[6] *Secure Shell* – mit kryptografischen Mitteln abgesicherte Variante der Remote Shell (RSH)

3 Analyse

Im Folgenden soll festgestellt werden, welche Anforderungen ein IPv6-Router erfüllen muss, damit er in einem ISP-Netzwerk eingesetzt werden kann. Zusätzlich werden die wichtigsten derzeit verfügbaren Alternativen für einen solchen Router vorgestellt.

3.1 Anforderungen an einen IPv6-Router

In der Regel besitzt ein Router mehr als eine physisches Netzwerk-Schnittstelle, z. B. Ethernet- oder ATM-Interfaces. In Ausnahmefällen steht nur eine solche Schnittstelle zur Verfügung. Häufig werden zur besseren Ausnutzung der wertvollen physischen Ports virtuelle Schnittstellen, wie *Virtual Channel* in ATM oder VLANs im Fall von Ethernet, benötigt.

Dem Tunneling von IPv6 über IPv4 kommt zwar eine abnehmende Bedeutung zu, da die native Unterstützung für IPv6 in den Backbones zunimmt, aber es wird in Einzelfällen nach wie vor benötigt. Insbesondere kann nicht jeder Kunde sämtliche seiner Router auf einen aktuellen Stand bringen und muss daher noch per Tunnel angeschlossen werden.

Neben der Fähigkeit, Pakete von einer Schnittstelle zur anderen zur transferieren, wird von modernen Routern heute auch die Fähigkeit zur Paket-Filterung erwartet. Mit dieser Funktionalität kann auch eine *Firewall* aufgebaut werden. Laut IPv6-Standards muss auch ein Router zur Absicherung der übertragenen Daten IPSec unterstützen. Diese Anforderung ist besonders wichtig, wenn *Mobile-IPv6* zum Einsatz kommt. Zum Aufbau von VPNs mittels IPSec werden üblicherweise dedizierte VPN-Terminatoren eingesetzt. Diese rentieren sich aber meist erst bei einer größeren Zahl von angebundenen Clients.

Zusätzlich muss der Router das IPv6-Feature *Stateless Address Autoconfiguration* unterstützen und entsprechende *Router-Advertisements* aussenden können. Neben diesem einfachen Verfahren zur Adress-Zuweisung an Clients ist auch die Unterstützung von *DHCPv6* vorzusehen. Dieses beinhaltet die Möglichkeit der Präfix-Delegation, also der Zuweisung eines Adress-Blocks an einen untergeordneten Router beim Kunden.

Um einen Router in ein Netzwerk eingliedern zu können, müssen von diesem neben dem statischen Routing auch dynamische Routing-Protokolle unterstützt werden. Für einfach aufgebaute Netze reicht als IGP oft das Protokoll RIPng, welches aber bei größeren Netzwerken nicht mehr praktikabel ist. ISP und Carrier setzen daher die Link-State-Protokolle OSPFv3 oder IS-IS[1] ein, um das Routing in ihren Netzwerken aufrecht zu erhalten.

Für ein ISP-Netzwerk, das Verbindungen zu mehreren ISPs und Carriern aufbauen möchte, ist der Einsatz des *Border-Gateway-Protokolls* BGP4 mit Multi-Protokoll-

[1] *Intermediate System to Intermediate System* – Routing-Protokoll der OSI-Netzwerke

Erweiterungen unerlässlich. Laut den IPv6-Standards ist es für Kunden-Netzwerke derzeit nötig, dass sie für Verbindungen zu mehreren Providern (Multihoming) mehrere IPv6-Präfixe besitzt. Dieses Vorgehen wird auch als eine wesentliche Hemmschwelle für die Einführung von IPv6 in Unternehmen gesehen. [KFH+99] Deshalb suchen die Mitglieder der *IPv6 Multihoming Working Group* aktuell nach effizienten Lösungen für das Multihoming.

3.2 Anforderungen eines Internet-Service-Providers

Im Umfeld eines mittelständischen ISP wie der *Logivison GmbH* werden Router mit unterschiedlichen Aufgaben betraut. Dabei gibt es drei wichtige Bereiche:

1. Anschluss von Kunden, die per Einwahl, DSL oder Standleitung Zugang zum Internet haben möchten.

2. Verdichtung der lokalen Netzwerke im Rechenzentrum und Weiterleitung zu den Außenanbindungen.

3. Außenanbindung des Provider-Netzwerks zu anderen Providern, vornehmlich an Internet-Austausch-Punkten sowie zu Carriern für die weltweite Erreichbarkeit.

Dabei werden vor allen Dingen Router benötigt, die mit Interface-Bandbreiten von 100 bis 155 $\frac{\text{MBit}}{\text{s}}$ effizient umgehen können. Meist werden dazu Ethernet- oder ATM-Schnittstellen gewählt, u. a. weil diese günstig anzuschaffen und zu unterhalten sind. Da das Hauptgeschäft eines solchen ISP im Anschluss lokaler Kunden und im Betrieb eines Rechenzentrums liegt, werden zusätzlich Schnittstellen für die gängigen Standleitungs-Typen benötigt.

Ein entscheidender Faktor eines Routers ist die Leistungsfähigkeit seiner *Forwarding-Engine*. Dies ist der Teil des Routers, der für die eigentliche Aufgabe des Weiterleitens eines Paketes zu einem anderen Interface zuständig ist. Das Bestreben der Router-Hersteller besteht darin, diesen Teil möglichst stark vom Hauptprozessor zu entkoppeln, um hohe Transferraten zu erreichen. Dazu werden häufig entsprechende Funktionen in sog. ASICs[2] ausgelagert.

Da diese Optimierungen vor allen Dingen für Netzwerk-Schnittstellen mit hohen Bandbreiten von Nöten sind, kann der Verzicht auf eine solche Beschleunigung durch den Einsatz einer entsprechend schnelleren CPU kompensiert werden, wenn dies die Architektur des Routers erlaubt.

Ein wichtiger Faktor bei der Entscheidung für einen bestimmten Router ist, neben den Fähigkeiten und dem Preis, auch das beim ISP vorhandene Know-How. Es ist nur schwer möglich, die Mitarbeiter auf verschiedene Arten von Konfigurationen, wie sie beim Einsatz von Routern unterschiedlicher Hersteller vorkommen, zu schulen. So ist meist das Wissen um Geräte von *Cisco Systems, Inc.* vorhanden, da *Cisco* einen sehr großen Marktanteil bei Netzwerk-Hardware besitzt.[Pet03] Auf Grund dieses hohen Markanteils hat sich auch ein lukrativer Gebrauchtmarkt für Cisco-Hardware entwickelt, so dass Router mittlerer Leistungsfähigkeit günstig aus 2. Hand zu erwerben sind.

[2] Application Specific Integrated Circuit – für eine spezielle Anwendung optimierter Schaltkreis

3.3 Derzeit verfügbare IPv6-Router

Generell lässt sich ein Router als fertiges Stück Hardware kaufen (sog. *integrierte Router*) oder aber mit Hilfe einer *Routing-Software* auf geeigneten Plattformen im Eigenbau umsetzen. Dieser Abschnitt stellt verschiedene Lösungen aus beiden Bereichen vor und vergleicht sie.

3.3.1 Integrierte Router

Die Hersteller von Routern bieten derzeit eine qualitativ sehr unterschiedliche Unterstützung von IPv6 für ihre Produkte an. Bei Marktbeobachtungen wurde festgestellt, dass neue Geräte häufig zumindest in der Lage sind die grundlegenden Anforderungen zu erfüllen. Ein Problem ergibt sich dagegen häufig beim Einsatz bestehender Hardware. Diese kann auf Grund der Architektur teilweise nicht um eine effiziente Unterstützung von IPv6-Forwarding ergänzt werden, oder der Hersteller stellt gar keine IPv6-Updates zur Verfügung.

In den meisten ISP-Netzwerken werden für IPv4 heute vor allen Dingen integrierte Router der Hersteller *Cisco* und *Juniper* eingesetzt, da diese die dortigen Leistungs-Anforderungen erfüllen. Daher wird im folgenden ein Blick in den Produktportfolio dieser beiden Hersteller geworfen.

Cisco Systems, Inc. [Cis] stellte als einer der ersten Hersteller IPv6-fähige Firmware für seine Router zur Verfügung und verfügt somit über einen Entwicklungsvorsprung. Bei mittelgroßen ISPs kommen hauptsächlich Modelle aus den Reihen 7200 und 7500 zum Einsatz. Das Design dieser Cisco-Router basiert auf einer zentralen CPU, die für das IPv6-Forwarding kaum Unterstützung durch zusätzliche ASICs erhält. Da die CPUs vergleichsweise langsam sind und sich auch um die Behandlung der Routing-Protokolle kümmern müssen, besteht bei diesen Geräten die Gefahr einer Denial-of-Service-Attacke (DoS), wenn das Weiterleiten der Pakete durch die CPU übernommen werden muss.

In Netzen mit höherem Bandbreiten-Bedarf kommt dagegen häufig die Baureihe 12000 zum Einsatz. Diese zeichnet sich durch eine starke Hardware-Unterstützung beim Forwarding und ein Multi-CPU-Design aus. Damit die Hardware-Beschleunigung sowohl für IPv4 und IPv6 zur Verfügung steht, benötigen diese Geräte allerdings aktuelle *Line-Cards* mit den Netzwerk-Interfaces. Wenn diese nicht vorhanden sind, gelten die gleichen Aussagen bezüglich DoS-Anfälligkeit wie bei den kleineren Modellen. Werden diese Router in einem MPLS-Backbone eingesetzt, müssen zumindest die Border-Router aufgerüstet werden, die die MPLS-Label vergeben. Dies kann zu hohen Investitionen für den Einsatz von IPv6 führen.

Eine Besonderheit des Betriebssystems der Cisco-Router (IOS – *Internet Operating System*) sind die vielen unterschiedlichen Versionen, die zeitgleich zur Verfügung stehen. Dies ermöglicht es dem Anwender einerseits die für seine Bedürfnisse passende Version auszuwählen, führt aber gleichzeitig dazu, dass er oft erst mehrere Versionen auf die korrekte Funktion der benötigten Features auf den eingesetzten Plattformen überprüfen muss. Der

Einsatz von IPv6 erfordert oft den Einsatz sog. *Technology*-Versionen, die qualitativ häufig Beta-Status haben und somit die Stabilität anderer Anwendungen negativ beeinflussen.

Da es einem kleineren ISP häufig an entsprechenden Testumgebungen mangelt und das Personal mit dem Betrieb des Produktiv-Netzes ausgelastet ist, ist die Einführung von IPv6 mit einigen Hürden belastet, obwohl der Hersteller behauptet, alle notwendigen Dinge zu unterstützen.

Juniper Networks, Inc. [Jun] besitzt nach Cisco den zweitgrößten Marktanteil an Internet-Routern und unterstützt seit der Version 5.1 seines JunOS Betriebssystems auch IPv6. Im Gegensatz zu Cisco verfolgt Juniper einen stark modularen Ansatz, was sich auch in der Software widerspiegelt. Zusätzlich ist die Hardware in eine *Routing*- und eine *Forwarding-Engine* unterteilt. Die Routing-Engine basiert dabei, je nach Modellreihe, auf einer Intel-Plattform mit einem stark abgewandelten BSD-Betriebssystem und verarbeitet die Routing-Protokolle bzw. erfüllt die Management-Aufgaben. Der eigentliche Paket-Transfer zwischen den Netzwerk-Interfaces wird von der Forwarding-Engine übernommen. Dies ist generell in voller Schnittstellen-Geschwindigkeit möglich.

Diese konsequente Zweiteilung erschwert es unter Umständen, neue Funktionen wie IPv6 in die Forwarding-Engine zu integrieren, da der schnelle Haupt-Prozessor für die Umstellungsphase nicht zur Weiterleitung genutzt werden kann. Aus diesem Grund ist eine IPv6-Unterstützung nur bei aktuellen Routern ohne Austausch möglich. Dafür verhindert dieses Design, dass durch das Forwarding der Pakete negative Effekte auf die Verarbeitung der Management-Aufgaben einwirken. Diese höhere Stabilität schlägt sich in einem hohen Preis für die Geräte nieder, zumal Juniper erst seit kurzer Zeit mit der Cisco 7200er Reihe vergleichbare und entsprechend günstige Router herstellt. Doch nur diese sind für einen mittelständischen ISP interessant.

3.3.2 Routing mittels Software

Neben den integrierten Routern mit eigener Hardware werden einige IPv6-fähige Software-Suiten zur Unterstützung des Routings angeboten. Diese lassen sich i. d. R. auf verschiedenen Hard- und Software-Plattformen benutzen. Eines der dabei am häufigsten unterstützen Betriebssysteme ist Linux. Daher werden hier einige dieser Programme kurz vorgestellt.

Quagga/GNU Zebra [Qua] ist eine der am häufigsten eingesetzte Routing-Software. Sie implementiert alle wichtigen Routing-Protokolle für IPv4 und IPv6. Da die Weiterentwicklung von *GNU Zebra* seit einigen Jahren nur noch schleppend voran geht, wird der offene Quellcode von einigen Programmierern unter dem Namen *Quagga* weitergepflegt.

Quagga wird auf Grund seiner ausgereiften Fähigkeiten auch gern als Route-Server an Internet-Austausch-Punkten wie dem DeCIX verwendet. Daher gilt zumindest seine IPv4-BGP-Implementation als sehr stabil. Die IPv6-Funktionalitäten erhalten zwar nicht so viel Aufmerksamkeit, ihnen wird von anderen Nutzern auf den Quagga-betreffenden Mailinglisten dennoch gute Qualität bescheinigt.

Quagga ist modular aufgebaut. Neben einem zentralen Server-Prozess, der die Kommunikation mit dem Betriebssystem-Kern übernimmt, können je nach Anforderung einzelne Prozesse für die unterschiedlichen Routing-Protokolle gestartet werden.

Einer der wichtigsten Punkte, die Zebra zu Popularität verhalfen, ist die Konfiguration mittels eines Kommandozeilen-Interfaces (CLI) mit Cisco-ähnlicher Syntax. Dieses ist zwar nicht im Zebra-Prozess zentralisiert, bietet aber dennoch einen leichten Einstieg, wenn das Cisco-CLI bereits bekannt ist. Da sich über diese Konfiguration nur die reinen Routing-Aufgaben bewältigen lassen, ist zusätzlich die Kenntnis und Konfiguration des darunter liegenden Betriebssystems von großer Bedeutung.

BIRD Internet Routing Daemon [FMM] ist eine weiteres Open-Source-Projekt, das sich zum Ziel gesetzt hat, eine vollständige Routing-Suite zu implementieren. Dabei setzen die Programmierer auf eine vollkommen eigene Konfigurations-Syntax, die an die Programmiersprache *C* angelehnt ist.

Die Unterstützung von IPv6 ist bereits vorhanden und umfasst BGP und RIPng. OSPFv3 wird zur Zeit noch nicht unterstützt.

NextHop GateD [Nex03] ist einer der ältesten Software-Router und wird vor allen Dingen zur Integration in *Embedded-Router* angeboten. Da es sich um ein kommerzielles Produkt handelt, ist die Entwicklung aller wichtigen Routing-Protokolle sehr weit fortgeschritten. Das schließt auch IPv6 ein.

GateD setzt statt verschiedener Prozesse für die unterschiedlichen Routing-Protokolle auf ein System mit kooperativem Multitasking, also einen monolithischen Ansatz. Dieser erleichtert die Anbindung eines zentralen CLIs. Somit lässt sich mit Hilfe von GateD auch eigene Router-Hardware bauen, ohne die Software komplett selbst entwickeln zu müssen.

Click Modular Router Project [Cli], das vornehmlich an den amerikanischen Universitäten MIT und UCLA entwickelt wird, beschreitet einen ganz anderen Weg. Ziel ist es eine eigene leicht erweiterbare Plattform für das Routing zu schaffen. Vor allen Dingen wird auf die effiziente Implementatierung des Forwardings geachtet. Um sich auf diesen Teil der Arbeit konzentrieren zu können, implementiert dieses Projekt einen entsprechenden Linux-Kernel-Treiber. Diese Plattform unterstützt auch IPv6-Forwarding und bietet somit die Möglichkeit, den Linux-eigenen Netzwerk-Code durch eine effizientere Variante zu ersetzen. Auf Grund des relativ frühen Entwicklungsstadiums wird zumindest heute von einem Einsatz im Regelbetrieb abgeraten.

4 Entwurf des Routers

Aus den analysierten Anforderungen lassen sich die folgenden Kriterien für den Linux-basierten IPv6-Router ableiten.

4.1 Pflichtenheft

4.1.1 Musskriterien

/M10/ Der Router nutzt Linux als Basis-Betriebssystem.

/M20/ Der Router soll sowohl IPv4- als auch IPv6-Forwarding unterstützen.

/M30/ Auf den Ethernet-Netzwerk-Schnittstellen sollen VLANs nach IEEE 802.1Q verwendet werden können.

/M40/ Als IGPs müssen für IPv4 RIPv2 und OSPFv2 sowie für IPv6 RIPng und OSPFv3 eingesetzt werden können.

/M50/ Als EGP muss BGP-4 mit Multi-Protokoll-Erweiterungen zur Unterstützung von IPv6 zur Verfügung stehen.

/M60/ IPv6-Tunnel zur Anbindung von Routern, die noch keine native IPv6-Anbindung besitzen, sollen hergestellt werden können.

/M70/ Eine Überwachung des Routers per SNMP muss ermöglicht werden.

/M80/ Logfiles sollen per Syslog über das Netz zu einem Syslog-Server weitergeleitet werden.

/M90/ Das System soll für Management-Zwecke nur über ein Netzwerk-Interface erreichbar sein, das keine externen Daten transportiert.

/M100/ Der Router soll unter Zuhilfenahme eines weiteren Routers eine erhöhte Ausfallsicherheit ermöglichen.

4.1.2 Wunschkriterien

/W10/ Die Terminierung von L2TP-Tunneln zum Anschluss von Kunden per ADSL soll möglich sein.

/W20/ Für die Authentifizierung der Einwahl soll RADIUS verwendet werden.

/W30/ MPLS-Basis-Funktionen sollten unterstützt werden.

/W40/ Die BGP-Verbindungen sollen per IPSec gesichert werden können.

/W50/ Die Konfiguration mit Hilfe eines seriellen Terminals soll ermöglicht werden.

/W60/ Die Systemzeit des Routers soll per *Network Time Protocol* (NTP) abgeglichen werden.

4.1.3 Abgrenzungskriterien

/A10/ Eine Beschleunigung des Paket-Forwardings durch zusätzliche Hardware ist nicht notwendig.

/A20/ Nur Ethernet-Schnittstellen müssen unterstützt werden.

/A30/ DHCPv6 wird derzeit nicht benötigt, da auf dem Markt noch keine Clients existieren.

4.1.4 Anwendungsbereiche

Als einzelnes Gerät ist ein solcher Router vor allen Dingen im Backbone oder zum Peering mit anderen ISP-Netzen an Internet-Knotenpunkten einzusetzen, da hierfür meist eine Redundanz durch weitere Leitungen und Router vorhanden ist.

Mit Hilfe mehrerer solcher Router kann auch die Anbindung von Kunden erfolgen, die per L2TP von einem entsprechenden Carrier übergeben werden. Prinzipbedingt wird beim Ausfall eines Routers vom LAC ein anderer aus dem Pool für die nächste Einwahl gewählt.

Auch der Einsatz als Router zum Anschluss von Ethernet-Segmenten in einem Rechenzentrum ist möglich. Allerdings ist zur Wahrung der Redundanz für IPv4 der Einsatz eines Hot-Standby-Protokolls wie Ciscos *Hot Standby Router Protocol* (HSRP) oder des offenen *Virtual Router Redundancy Protocol*s (VRRP) notwendig. Für IPv6 sind hierfür i. d. R. keine weiteren Maßnahmen erforderlich.

4.2 Notwendige Software

Um die o. g. Kriterien erfüllen zu können, müssen verschiedene Software-Pakete installiert werden. Die zu diesem Zweck ausgewählten Programme werden hier mit ihrer Bezugsquelle aufgeführt. Soweit möglich werden diese aber als Binär-Paket aus der verwendeten Linux-Distribution installiert.

- Debian GNU/Linux – `http://www.debian.org/` – als Basis-Betriebssystem mit einem Linux Kernel 2.6.x,
- VLAN Tools – `http://www.candelatech.com/~greear/vlan.html` – zur Konfiguration von VLANs,
- Quagga – `http://www.quagga.net/` – als Routing-Daemon,
- Net-SNMP – `http://www.net-snmp.org/` – bietet die Funktion als SNMP-Agent zu arbeiten und kann mit Hilfe einer SMUX-Schnittstelle erweitert werden,
- l2tpd – `http://www.l2tpd.org/` – wird in Zusammenarbeit mit PPP zur Terminierung der L2TP-Tunnel benutzt,
- IPSec-Tools – `http://ipsec-tools.sourceforge.net/` – beinhalten die Werkzeuge, um die IPSec-Unterstützung des Kernel zu konfigurieren,
- MPLS for Linux – `http://mpls-linux.sourceforge.net/` – die aktuell am weitesten fortgeschrittene Implementation von MPLS unter Linux, erfordert teilweise die Integration in Quagga,
- NTP – `http://www.ntp.org/` – dient zur Zeit-Synchronisation.

5 Implementation und Integration

Als Basis-System fiel die Auswahl auf Debian GNU/Linux als Betriebssystem-Distribution, da diese auch sonst häufig im Umfeld der Firma *Logivision* zum Einsatz kommt und daher genügend Know-How zur Administration zur Verfügung steht.

Die Bearbeitung der Routing-Protokolle soll durch *Quagga/Zebra* erfolgen, weil es eine Cisco-ähnliche Bedienung bietet. Außerdem unterstützt es alle notwendigen Routing-Protokolle. Der Support und die Qualität genügen den Ansprüchen.

5.1 Grundsystem

Debian bietet verschiedene Zweige seiner Distribution an, die eine unterschiedlich aktuelle Software-Auswahl zur Verfügung stellen. Der *stable*-Zweig enthält nur Software, die lange genug getestet wurde. Daher ist ein großer Teil dieser Software derzeit ca. zwei Jahre alt. Im *unstable*-Zweig befinden sich Pakete, die dem aktuellen Stand der Entwicklung entsprechen, dafür aber nur wenig getestet wurden. Aus diesem Zweig werden Pakete, die bestimmten Qualitäts-Anforderungen gerecht werden, in den *testing*-Zweig übernommen, welcher in Zukunft zu *stable* werden soll. [deb03]

Die Entscheidung fiel auf die *unstable*-Distribution, da für die Bereitstellung der geforderten Funktionen teilweise aktuelle Software benötigt wird, die nur in diesem Zweig der Distribution erhältlich ist. Da sich Tests in virtuellen Maschinen sehr leicht durchführen lassen, kann die notwendige Qualität trotzdem sichergestellt werden.

5.1.1 Installation des Debian GNU/Linux-Basis-System

Zur Installation von *Debian unstable* muss ein Umweg über die Installation eines *stable*- oder *testing*-Grundsystems und anschließendem Upgrade auf *unstable* gegangen werden. Dazu wird im einfachsten Fall ein CD-Image von `http://www.debian.org/CD/netinst/` heruntergeladen, auf eine CD gebrannt, von dieser gebootet und der Menüführung gefolgt, bis eine Basis-Installation auf der Festplatte vorliegt. Eine zusätzliche Paket-Auswahl muss zu diesem Zeitpunkt nicht erfolgen.

Nachdem das Basis-System installiert ist, erfolgt das Upgrade auf *unstable*. Zu diesem Zweck muss die Datei `/etc/apt/sources.list` wie in Listing 5.1 beschrieben geändert werden. Daraufhin ist ein Update der Paketliste und das eigentliche Upgrade durchzuführen. Die dafür notwendige Befehlsfolge zeigt Listing 5.2.

Wenn das Upgrade erfolgt ist, sollten die im späteren Verlauf benötigten Pakete installiert und für den Betrieb Unwichtiges gelöscht werden. Der Befehl `apt-get install <Paket>` installiert das jeweilige Paket und alle anderen dafür notwendigen Pakete.

```
deb http://ftp.de.debian.org/debian/ unstable main contrib
deb http://debian.seabone.net/debian-ipv6 sid ipv6
```

Listing 5.1: `/etc/apt/sources.list` für ein Debian Unstable

```
apt-get update
apt-get dist-upgrade
```

Listing 5.2: Befehle zum Upgrade der Distribution

Auf diese Weise sollten nun die folgenden Pakete hinzugefügt werden:

- `ssh` – SSH-Server zur sicheren Fernwartung,
- `iproute` – Programme zur Unterstützung der erweiterten IP-Routing-Funktionalitäten aktueller Linux-Kernel – für den Aufbau von IPv6-Tunneln notwendig,
- `vlan` – Unterstützung für VLANs an Ethernet-Schnittstellen,
- `ntp-simple` – ein NTP-Server, der zur Aktualisierung der System-Zeit per Netzwerk ausreichend ist.

5.1.2 Einstellungen des Grundsystems

Das so installierte Linux-Basis-System benötigt einige Veränderungen, um den Anforderungen zu genügen, die an ein System gestellt werden, das eine möglichst hohe Verfügbarkeit aufweisen soll.

Für eine erhöhte Sicherheit des Dateisystems bei Stromausfall sollte ein Journaling-Filesystem zum Einsatz kommen. Das Dateisystem *Ext3* hat sich hierbei besonders bewährt. Nach Möglichkeit sollte dies bereits bei der Installation geschehen. Diese Veränderung kann aber auch nachträglich an einer vorhandenen *Ext2*-Partition vorgenommen werden. In den meisten Fällen ist es nicht notwendig, dass nach einer bestimmten Anzahl von Neustarts des Systems das Filesystem auf Fehler überprüft wird. Um diese Funktion zu deaktivieren, sind die in Listing 5.3 aufgeführten Schritte nötig. Es wird hier von einer Installation auf der Partition `/dev/hda1` ausgegangen.

```
init 1
mount -o remount,ro /dev/hda1 /
e2fsck -f /dev/hda1
tune2fs -j /dev/hda1  # <-- Wenn eine Umwandlung in Ext3 noetig ist
tune2fs -c 0 -i 0 /dev/hda1
mount -o remount,rw /dev/hda1 /
init 2
```

Listing 5.3: Tuning der Filesystem-Parameter

Standardmäßig ist ein Login aus der Ferne auf einem Debian-System erst möglich, nachdem alle Dienste gestartet wurden. Sollte es beim Start zu Schwierigkeiten kommen, kann dieses Verhalten dazu führen, dass physischer Zugang zum Rechner nötig ist, was oft mit großem Aufwand verbunden ist. Daher sollte diese Funktion deaktiviert werden. Dazu muss in der Datei `/etc/default/rcS` folgende Einstellung vorgenommen werden:

```
DELAYLOGIN=no
```

Sollte doch einmal lokal auf den Rechner zugegriffen werden müssen, weil keine Netzwerk-Verbindung zur Verfügung steht, ist es selten praktikabel eine Tastatur und einen Monitor anzuschließen. Da auch andere Netzwerk-Hardware nur einen Zugriff per serieller Konsole ermöglichen, steht meist ein solches Terminal oder ein Notebook mit serieller Schnittstelle zur Verfügung. Deshalb bietet es sich an, auch den Linux-basierten Router per seriellem Terminal bedienen zu können.

Zu diesem Zweck wird auf einem Unix-System üblicherweise ein *getty* gestartet, der auf einen Verbindungsaufbau an seriellen Schnittstellen wartet. Das folgende Beispiel zeigt die entsprechende Konfiguration in der Datei `/etc/inittab`, die auch mit minimal beschalteten seriellen Kabeln an der ersten seriellen Schnittstelle arbeitet:

```
T0:23:respawn:/sbin/getty -L ttyS0 9600 vt100
```

Soll eine andere serielle Schnittstelle zum Einsatz kommen, ist standardmäßig ein Login als der Superuser `root` nicht möglich. Um dies zu ändern, muss der Name des entsprechenden Gerätes (im obigen Beispiel `ttyS0`) in die Datei `/etc/securetty` eingetragen werden.

Mit Hilfe dieses Mechanismus kann auch ein Modem oder ISDN-Terminal-Adapter angeschlossen werden, um eine Fernbedienung auch ohne Netzwerk-Verbindung zu ermöglichen.

Mit den vorgenannten Einstellungen ist der Zugriff auf den Router erst möglich, wenn das Betriebssystem erfolgreich geladen wurde. Um auch Fehler während des Boot-Prozesses auf dem seriellen Terminal erkennen zu können, bieten sowohl der Boot-Loader *LILO* als auch der Linux-Kernel die Möglichkeit ein Terminal anzusteuern. Zu diesem Zweck sind die in Listing 5.4 aufgeführten Einstellungen in der Datei `/etc/lilo.conf` vorzunehmen.

```
serial=0,9600n8
append=" console=tty0 console=ttyS0,9600n8 "
```

Listing 5.4: Konfiguration des *LILO* zur Steuerung serieller Terminals

Für das Debugging in Netzwerken ist es oft sinnvoll eine einheitliche Zeitbasis zu Grunde legen zu können. Eine zuverlässige Möglichkeit zur Synchronisation der Zeit zwischen Computern mit IP-Vernetzung bietet das *Network Time Protocol* (NTP). Listing 5.5 stellt ein Konfigurations-Beispiel für die Datei `/etc/ntp.conf` dar. Zeile 2 zeigt die Verwendung von per Multicast erreichbaren NTP-Servern innerhalb der zugehörigen Organisation.

Eine gleichberechtigte Beziehung zwischen zwei NTP-fähigen Routern zeigt Zeile 3. Soll ein dedizierter NTP-Server verwendet werden, kommt Zeile 4 zum Einsatz.

```
driftfile /var/lib/ntp/ntp.drift
manycastclient ff05::101
peer routerB.firma.bsp
server ntp.firma.bsp
```

Listing 5.5: Beispiel `/etc/ntp.conf`

Die auf einem Unix-System auflaufenden Log-Meldungen werden standardmäßig vom `syslog`-Daemon in Log-Dateien geschrieben. Dieses Vorgehen ist bei einem System mit wenig Festspeicher ungünstig, da dieser sehr schnell an seine Grenzen stößt. Außerdem ist eine dezentrale Speicherung und Auswertung der Meldungen nur schwer durchzuführen, was das Auffinden von Fehlern unnötig verkompliziert. Daher bietet der `syslogd` die Möglichkeit, Log-Meldungen per UDP an einen Syslog-Server zu senden.

Listing 5.6 zeigt, wie alle Meldungen an den Server `syslog.firma.bsp` weitergeleitet werden. Um bei einem Ausfall der DNS-Namens-Auflösung die korrekte Funktion sicherzustellen, sollte dieser Hostname mit der zugehörigen IP-Adresse in die Datei `/etc/hosts` eingetragen werden. Zusätzlich werden in diesem Beispiel Meldungen ab der Priorität *kritisch* (`crit`) an alle angemeldeten Benutzer versandt, so dass Fehler sofort bemerkt werden können.

```
*.*                     @syslog.firma.bsp
*.crit                  *
```

Listing 5.6: `/etc/syslog.conf` zur Nutzung eines Syslog-Servers

Es ist empfehlenswert, einen unprivilegierter Nutzer anzulegen. Wie bei jedem Unix-System besitzt der Super-User `root` das Recht beliebige Änderungen vorzunehmen. Mit dem Unix-Rechte-System lassen sich somit die Auswirkungen einer etwaigen Kompromittierung minimieren. Beim Anlegen des Benutzers `admin` mit dem Befehl

```
adduser admin
```

werden die Angabe eines Passwortes und zusätzlicher Informationen gefordert. Es gelten die allgemeinen Regeln für sichere Passworte.

5.1.3 Kernel-Konfiguration

Auch wenn *Debian* fertige Kernel-Pakete mit allen für den Betrieb eines Routers notwendigen Optionen zur Verfügung stellt, ist es nach wie vor sinnvoll einen auf die eigenen Bedürfnisse abgestimmten Kernel einzusetzen. Eine gute Integration des Kernels in die Distribution ist in der Regel nur unter Einbeziehung der mitgelieferten Hilfsmittel zu erreichen. Diese Werkzeuge ermöglichen es auch, den Kernel auf einem anderen Rechner zu compilieren. Dieses Vorgehen ist vor Allem notwendig, wenn der Datenträger des Routers nur begrenzten Platz bietet, z. B. weil eine Flash-Disk zum Einsatz kommt.

Da der originale Linux-Kernel derzeit bezüglich einiger Details der IPv6-Standards nicht auf dem aktuellen Stand ist, lohnt es sich die Patches des *USAGI-Projektes* [USA] einzusetzen. Neben den Anpassungen an den aktuellen Stand der Standardisierung des ND-Protokolls, bieten sie auch Connection-Tracking für den IPv6-Paket-Filter.

Um ein Debian-Paket für einen (zum Zeitpunkt des Schreibens dieser Arbeit aktuellen) Linux Kernel 2.6.6 zu erstellen, sind die in Listing 5.7 dargestellten Befehle einzugeben. Mit der ersten Zeile wird der Source-Code als Debian-Paket heruntergeladen, der mit dem Befehl der dritten Zeile entpackt wird. Zusätzlich wird ein USAGI-Patch heruntergeladen und in Zeile 6 appliziert. Nachdem die Konfiguration des Kernels mit dem Befehl der siebten Zeile vorgenommen wurde, erstellt die achte Zeile das eigentliche Paket mit dem Kernel und den Modulen im Verzeichnis `/usr/src`. Das so entstandene Paket kann nun auf dem Zielsystem mit dem Paket-Manager `dpkg` installiert werden.

```
apt-get install kernel-source-2.6.6
cd /usr/src/
tar xjf kernel-source-2.6.6.tar.bz2
wget ftp://ftp.linux-ipv6.org/pub/usagi/snap/split/usagi-linux26- ↵
   s20040524-2.6.6.diff.bz2
cd kernel-source-2.6.6
bzcat ../usagi-linux26-s20040524-2.6.6.diff.bz2 | patch -p1
make menuconfig
make-kpkg kernel_image
```

Listing 5.7: Erstellen eines Kernel-Paketes

Der Konfiguration des Kernels kommt dabei eine besondere Bedeutung zu. Es ist darauf zu achten, dass die Treiber für alle notwendigen Hardware-Komponenten fest in den Kernel eingebaut werden. Dies Vorgehen erspart das spätere Nachladen als Modul und vermeidet somit Probleme, die im Umgang mit Modulen teilweise auftreten. Die Benutzung der IPv6-Unterstützung als Modul führte z. B. im Verlauf der praktischen Tests unter noch unklaren Bedingungen zu einem stetig anwachsenden Speicher-Verbrauch.

Für einen Router sind vor allen Dingen die Netzwerk-Parameter wichtig. Diese befinden sich innerhalb der Menüstruktur unter *Device Drivers Networking Support Networking options*. Die folgenden Optionen sind für die korrekte Funktion als Router unbedingt zu aktivieren:

- *IP: advanced router*
- *IP: equal cost multipath*
- *IP: tunneling*
- *IP: GRE tunnels over IP*
- *IP: TCP syncookie support*
- *The IPv6 protocol*

- *Network packet filtering* inkl. der darunter befindlichen Module für IPv4- und IPv6-Paket-Filterung
- *802.1Q VLAN Support*
- die Optionen für IPSec, AH, ESP und IPComp

Darüber hinaus ist die Aktivierung der Unterstützung der eingesetzten Netzwerk-Schnittstellen nötig. Neben den physischen Schnittstellen, wie Ethernet und ATM, muss für den Einsatz von Einwahl und L2TP-Tunneln auch PPP aktiviert sein.

Bei der Auswahl der einzusetzenden Netzwerk-Karten und -Treiber sollte darauf geachtet werden, dass diese auch größere Ethernet-Frames unterstützen, da der Einsatz von VLANs mit einer MTU von weniger als 1500 Byte sonst nicht sinnvoll möglich ist. Für einige Treiber stehen entsprechende Patches zur Verfügung. Der im Beispiel-System eingesetzte `e100`-Treiber für Intel Fast-Ethernet NICs benötigt keine zusätzlichen Patches.

Eine komplette Konfigurations-Datei für einen funktionsfähigen Kernel ist auf dem beigefügten Datenträger enthalten (siehe Anhang C).

5.1.4 Netzwerk-Konfiguration

Während der Installation wird normalerweise eine automatische Konfiguration der Netzwerk-Parameter per DHCP vorgenommen. Für wichtige Teile der Infrastruktur wie einen Router ist dieses Vorgehen allerdings ungeeignet. Einstellungen, die auch unabhängig von der Funktion anderer Systeme im Netzwerk sein sollen, müssen manuell vorgenommen werden.

Vor allen Dingen ist es notwendig die einzelnen Netzwerk-Schnittstellen zu konfigurieren. Dazu gehört sowohl die Einstellung von evtl. nötigen Hardware-Parametern, als auch die Zuweisung von IP-Adressen. Zur Verwendung virtueller Interfaces, etwa IPv6-Tunneln, müssen diese dem Kernel zusätzlich bekannt gemacht werden.

Neben der Möglichkeit die dafür notwendigen Befehle mit Hilfe eines eigenen Skriptes beim System-Start abarbeiten zu lassen, bietet Debian einen komfortablen und einheitlichen Mechanismus zur Konfiguration der Netzwerk-Interfaces an. Alle nötigen Einstellungen werden dazu in der Datei `/etc/network/interfaces` vorgenommen. Die dort getroffenen Einstellungen werden beim Systemstart abgearbeitet und ermöglichen auch ein manuelles Aktivieren und Deaktivieren eines Interfaces mit Hilfe der Befehle `ifup` und `ifdown`.

Das Beispiel in Listing 5.8 zeigt die grundlegende Konfiguration des Interfaces `eth0`. Die beim Systemstart automatisch zu konfigurierenden Schnittstellen werden wie in Zeile 1 hinter einem `auto`-Schlüsselwort aufgeführt. In folgenden Beispielen wird zur Vereinfachung darauf verzichtet. Für jede Adress-Familie (`inet` und `inet6`) muss ein einzelner `iface`-Eintrag erfolgen. Hier werden der Schnittstelle die IPv4-Adresse `192.168.8.10` im Netz `192.168.8.0/24` und die IPv6-Adresse `2001:db8:10:1::10/64` zugewiesen. Dies stellt eine minimale Konfiguration dar, die nach Bedarf erweitert werden kann.


```
1 auto eth0
2
3 iface eth0 inet static
4         address 192.168.8.10
5         netmask 255.255.255.0
6
7 iface eth0 inet6 static
8         address 2001:db8:10:1::10
9         netmask 64
```

Listing 5.8: Konfiguration einer Ethernet-Schnittstelle

Das Programm `vconfig` ermöglicht die Verwaltung von 802.1Q VLANs unter Linux. Ein damit verwaltetes virtuelles Interface muss vor dem Parametrisieren erst erzeugt werden. Diese Arbeit nehmen die Skripte aus dem Debian-Paket `vlan` ab und bieten so eine nahtlose Integration in die zentrale Netzwerk-Konfiguration an. Listing 5.9 zeigt das Anlegen des VLANs mit der VLAN-ID 10 auf der Schnittstelle `eth0`. Für eine korrekte Funktion ist die Konfiguration der Gegenseite (i. d. R. ein Switch) notwendig.

```
1 iface eth0.10 inet6 static
2         address 2001:db8:100:10::10
3         netmask 64
```

Listing 5.9: Konfiguration eines VLANs

Um lediglich das VLAN-Interface anzulegen, die Einstellungen aber später auf andere Weise vorzunehmen, wird die in Listing 5.10 gezeigte Methode `manual` angeboten. Dieses Vorgehen ist insbesondere dann zu empfehlen, wenn möglichst große Teile der Konfiguration durch die Mechanismen des Routing-Daemons verwalten werden sollen, wie dies bei Quagga möglich ist.

```
1 iface eth0.10 inet manual
```

Listing 5.10: Anlegen eines VLANs mit manueller Konfiguration

Für den Aufbau eines IPv6-in-IPv4-Tunnels wird ein Interface mit der Methode `v4tunnel` verwendet. Zusätzlich zu den obigen Einstellungen werden, wie in Listing 5.11 gezeigt, die IPv4-Adressen der beiden Endpunkte für die Enkapsulierung benötigt. Wenn nicht die minimale MTU von 1280 Bytes verwendet werden soll, weil die dazwischen liegenden Übertragungswege größere Pakete zulassen, kann dies auch angegeben werden.

Neben den Einstellungen der Netzwerk-Schnittstellen ist es für den Betrieb sinnvoll, die Namens-Auflösung per DNS zu ermöglichen. Zu diesem Zweck werden in der Datei `/etc/resolv.conf` die IP-Adressen der verfügbaren DNS-Server eingetragen, wie es Listing 5.12 zeigt.

Da ein Router für gewöhnlich alle Informationen für seine Routing-Tabelle aus den Routing-Protokollen gewinnt, ist auf die Definition eines *Default-Gateways* mit Hilfe

```
iface 6tunnel inet6 v4tunnel
        address 2001:db8:10:3::1
        netmask 64
        local 192.168.8.10
        endpoint 192.168.1.1
        mtu 1480
```

Listing 5.11: Konfiguration eines IPv6-Tunnels

```
search firma.bsp
nameserver 2001:db8:100::1
nameserver 2001:db8:200::1
nameserver 192.168.8.2
```

Listing 5.12: Beispiel für `/etc/resolv.conf`

des `gateway`-Schlüsselwortes zu verzichten. Um im Falle eines Ausfalls der Routing-Protokolle dennoch per Netzwerk auf das System zugreifen zu können, ist es nötig mehrere direkt verbundene Zwischenstationen (z. B. andere Router) zu nutzen. Alternativ kann zumindest eine statische Route in Richtung des Management-Netzwerks definiert werden.

Zur Konfiguration wichtiger statischer Routen werden die vorhandenen Erweiterungs-Mechanismen der Netzwerk-Konfiguration genutzt. Diese ermöglichen es u. a. nach der Konfiguration eines Interfaces bestimmte Befehle auszuführen. Listing 5.13 zeigt die Erweiterung des obigen Beispiels um eine statische Route zum Management-Netz `2001:db8:1::/64` über einen anderen im lokalen Netz befindlichen Router mit der Adresse `2001:db8:10:1::1`.

```
iface eth0 inet6 static
        address 2001:db8:10:1::10
        netmask 64
        up /sbin/ip route add 2001:db8:1::/64 gw via 2001:db8:10:1::1
```

Listing 5.13: Konfiguration von statischen Routen

5.1.5 Beschränkung des Netzwerk-Zugriffs

Um den Router vor Angriffen von außen zu schützen, wie es /M90/ fordert, müssen einige Maßnahmen ergriffen werden. Dazu gilt es einerseits, nur die notwendigen Dienste zu aktivieren, und andererseits entsprechende Paket-Filter-Regeln zu definieren. Da ein Router i. d. R. nicht wie ein Server durch eine Firewall geschützt werden kann, bedarf es einer lokalen Installation der Filter.

In einem Debian-Standard-System wird auf Grund von Abhängigkeiten zwischen den Paketen der Super-Server `inetd` installiert, der in der Ausgangs-Konfiguration einige für den Betrieb nicht notwendige Dienste zur Verfügung stellt. Um dies zu verhindern, muss der

Start des Dienstes deaktiviert werden, indem das folgende Kommando als `root` ausgeführt wird:

```
update-rc.d -f inetd remove
```

Alternativ können die Abhängigkeiten auch durch Erstellen eines Dummy-Paketes erfüllt werden und das eigentliche `inetd`-Paket deinstalliert werden.

Statt der Möglichkeit mittels *TCP Wrapper* eine umfassende Zugriffs-Steuerung auf Dienst-Ebene durch Konfiguration der Dateien `/etc/hosts.allow` und `hosts.deny` vorzunehmen, kann mit Hilfe eines Paket-Filters in tieferen Schichten des OSI-RM angesetzt werden. Diesem Zweck dient in *Linux* das *Netfilter*-System, welches mit Hilfe der Befehle `iptables` (IPv4) sowie `ip6tables` (IPv6) konfiguriert wird.

Je Protokoll-Familie existieren dabei Ketten mit Filter-Regeln, die nacheinander geprüft werden. Jede Regel besteht dabei aus einem Teil, der mit einem inspizierten Paket verglichen wird und einem Befehl, was mit diesem Paket geschehen soll (dieser wird auch „Ziel“ genannt). Normalerweise wird nur der Befehl der ersten zutreffenden Regel ausgeführt. Neben der Möglichkeit eigene Regel-Ketten zu definieren, die wiederum als Ziele dienen können, existieren die folgenden vordefinierten Ketten:

- `INPUT` – prüft die für das lokale System bestimmten Pakete,
- `OUTPUT` – prüft die vom lokalen System ausgehenden Pakete,
- `FORWARD` – prüft die nicht für das lokale System bestimmten, aber durchlaufenden Pakete.

Für die Abschottung des Routers vor Zugriffen von außen ist vor allen Dingen die Kette `INPUT` zu beachten. Der eingesetzte Regel-Satz sorgt dafür, dass standardmäßig alle Verbindungs-Versuche abgelehnt und nur bestimmte Pakete zugelassen werden. Dazu gehören lokale Verbindungen und solche aus dem physisch abgetrennten Management-Netz. Des Weiteren müssen auf den nach außen gewandten Schnittstellen auch etwaige Routing-Protokolle zugelassen werden. Auch die für das Basis-System notwendigen Dienste wie DNS und NTP müssen mit entsprechenden Regeln explizit zugelassen sein.

Listing 5.14 zeigt beispielhaft, welche Filter-Regeln notwendig sind, um eine korrekte Funktion des Routers bei gutem Schutz zu ermöglichen. Das Skript ist in der für Start-Skripte üblichen Form verfasst, so dass vor allen Dingen die Parameter `start` und `stop` akzeptiert werden. Damit das Skript beim System-Start ausgeführt wird, muss es im Verzeichnis `/etc/init.d` z. B. als `pktfilter` abgelegt und anschließend aktiviert werden. Dies kann bei dem von Debian verwendeten System-V-Init-System durch den Befehl

```
update-rc.d pktfilter start 01 2 3 4 5 .
```

erfolgen. Dadurch wird eine frühzeitige Ausführung (Priorität 01) in den Runlevels, die für den späteren Betrieb in Frage kommen (2, 3, 4 und 5), sichergestellt. Ein automatisches Ausführen der Stop-Regel ist für einen dauerhaften Einsatz nicht nötig.

```
#!/bin/sh
NAME=pktfilter
IP6T=/sbin/ip6tables
IPT=/sbin/iptables

set -e

case "$1" in
  start|restart|force-reload)
        # Regel-Ketten leeren
        $IPT -F INPUT
        $IP6T -F INPUT

        # lokale Zugriffe und vom Management-Netz an eth0 zulassen
        $IPT -A INPUT -i lo -j ACCEPT
        $IP6T -A INPUT -i lo -j ACCEPT
        $IPT -A INPUT -i eth0 -j ACCEPT
        $IP6T -A INPUT -i eth0 -j ACCEPT

        # statt Default-Regel besser explizite DROP-Regel verwenden,
        # damit nach einem Leeren der Regeln weiterhin Zugriff
        # moeglich ist
        $IPT -A INPUT -j DROP
        $IP6T -A INPUT -j DROP

        # L2TP-Verbindung von den moeglichen LACs zulassen
        $IPT -I INPUT -s 10.9.8.7/24 -p udp --dport 1701 -j ACCEPT

        # Verbindungen von/zu Link-lokalen Adressen sind wichtig
        # diese Regeln sollten evtl. konkreter gefasst werden
        $IP6T -I INPUT -s fe80::/64 -j ACCEPT
        $IPT -I INPUT -d 224.0.0.0/24 -j ACCEPT

        # BGP-Sessions auch auf anderen Interfaces von den Peers
        # annehmen -- evtl. naeher spezifizieren
        $IPT -I INPUT -p tcp --dport 179 -j ACCEPT
        $IP6T -I INPUT -p tcp --dport 179 -j ACCEPT

        # ICMP ist fuer die korrekte Funktion unerlaesslich
        $IPT -I INPUT -p icmp -j ACCEPT
        $IP6T -I INPUT -p ipv6-icmp -j ACCEPT

        # selbst aufgebaute TCP-Sessions (nicht nur SYN-Flag) zulassen
        $IPT -I INPUT -p tcp ! --syn -j ACCEPT
        $IP6T -I INPUT -p tcp ! --syn -j ACCEPT

        # DNS und NTP sollen von den eigenen Servern zugelassen
        # werden, soweit nicht im Management-LAN
        $IPT -I INPUT -p udp -s 10.0.8.0/24 --sport 53 -j ACCEPT
        $IP6T -I INPUT -p udp -s 2001:db8:8::/64 --sport 53 -j ACCEPT
        $IPT -I INPUT -p udp -s 10.0.10.0/24 --dport 123 -j ACCEPT
        $IP6T -I INPUT -p udp -s 2001:db8:8::/64 --dport 123 -j ACCEPT
```



```
        # getunnelte IPv6-Pakete zulassen
        $IPT -I INPUT -p ipv6 -j ACCEPT

        # IPSec-Pakete (Protokolle AH, ESP), IKE zulassen
        $IPT -I INPUT -p udp --sport 500 --dport 500 -j ACCEPT
        $IP6T -I INPUT -p udp --sport 500 --dport 500 -j ACCEPT
        $IPT -I INPUT -p ah -j ACCEPT
        $IP6T -I INPUT -p ah -j ACCEPT
        $IPT -I INPUT -p esp -j ACCEPT
        $IP6T -I INPUT -p esp -j ACCEPT
        ;;
  stop)
        # Regel-Ketten leeren
        $IPT -F INPUT
        $IP6T -F INPUT
        ;;
  *)
        echo "Usage: /etc/init.d/$NAME {start|stop|restart|force- ↵
           reload}" >&2
        exit 1
        ;;
esac

exit 0
```

Listing 5.14: Beispiel-Skript für den Paket-Filter

5.1.6 Einstellungen für das Routing

Nachdem ein stabiles Basis-System geschaffen wurde, bedarf es einiger spezieller Einstellungen, damit das System auch als Router eingesetzt werden kann.

Die Haupt-Funktion eines Routers, das Weiterleiten von Paketen, bedarf expliziter Aktivierung. Diese Einstellung kann bei modernen Linux-Kerneln auf einzelne Netzwerk-Schnittstellen beschränkt werden. Dies bedeutet aber nicht, dass für die Trennung von Netzen auf entsprechende Paket-Filter-Regeln verzichten werden kann.

Zur Konfiguration dieses Kernel-Parameters stehen zwei Wege zur Verfügung. Traditionell kann er über das `/proc`-Dateisystem erreicht werden. Um zu anderen Systemen (insbesondere BSD) kompatibler zu sein, stellt das Kommando `sysctl` eine Schnittstelle zur Verfügung, die die Parameter beim System-Start verändern kann. Zu diesem Zweck werden die gewünschten Einstellungen in der Datei `/etc/sysctl.conf` in der Form `<Parameter>=<Wert>` abgelegt.

Zur korrekten Funktion der in Listing 5.15 aufgeführten Einstellungen ist es unerlässlich, dass die IPv6-Unterstützung fest in den Kernel eingebunden und nicht als Modul vorliegt, da diese Änderungen vor dem Laden von Modulen ausgeführt werden. Neben der Aktivierung der Forwardings ist es für IPv4 i. d. R. notwendig den *Reverse Path Filter* zu deaktivieren. Ist dieser aktiv, werden Pakete von Absendern abgewiesen, die laut Routing-Tabelle nicht

über das selbe Interface erreicht werden. Außerdem ist jegliche automatische Konfiguration zu deaktivieren.

```
net.ipv4.conf.all.forwarding=1
net.ipv4.conf.all.rp_filter=0
net.ipv6.conf.all.forwarding=1
net.ipv6.conf.all.autoconf=0
```

Listing 5.15: Notwendige `sysctl`-Parameter für das Routing

Nachdem diese Einstellungen somit für alle Interfaces vorgenommen wurden, kann das Verhalten einzelner Schnittstellen nach Bedarf geändert werden.

Außerdem bietet Debian die Datei `/etc/network/options` zur Konfiguration dieser Parameter an, deren Änderung sich allerdings nur auf IPv4 auswirken. Daher muss die Abarbeitung dieser Einstellungen entweder deaktiviert oder entsprechend angepasst werden. Dazu brauchen lediglich die Zeilen

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

eingefügt werden.

5.2 Dynamisches Routing mit Quagga

Das aus GNU/Zebra hervorgegangene Programm-Paket Quagga stellt eine freie Implementation aller für den Internet-Bereich notwendigen Routing-Protokolle zur Verfügung. Das *InterOperability Laboratory* an der *Universität New Hampshire* hat deren Funktion für IPv4 getestet und größtenteils als standardkonform zertifiziert.

Quagga basiert auf einem verteilten Ansatz. Die Bearbeitung der verschiedenen Routing-Protokolle wird von verschiedenen Daemons übernommen, die mit dem zentralen Daemon `zebra` per Unix-Domain-Sockets kommunizieren. Dieser Ansatz soll zu einer höheren Stabilität führen und ermöglicht die einfache Erweiterbarkeit der Routing-Suite.

5.2.1 Installation

Quagga kann einerseits aus dem Quellcode selbst compiliert werden oder in Form des Debian-Paketes `quagga` installiert werden. Die letztgenannte Variante bietet den Vorteil, dass das Paket bereits alle notwendigen Optionen enthält und einen ausgereiften Mechanismus zum Start der einzelnen Daemons zur Verfügung stellt.

Nach der Installation des Pakets mittels

```
apt-get install quagga
```

ist ein Start des Routing-Daemons noch nicht möglich. Zuerst muss angegeben werden, welche Routing-Protokolle zum Einsatz kommen sollen. In der Datei `/etc/quagga/daemons` kann pro zu startendem Daemon eine Priorität für die Start-Reihenfolge angegeben werden. Um nur den Haupt-Prozess `zebra` zu starten, kann das in Listing 5.16 aufgeführte Beispiel genutzt werden. Zusätzliche Routing-Protokolle werden in den folgenden Abschnitten dieses Kapitels behandelt.

```
zebra=1
bgpd=no
ospfd=no
ospf6d=no
ripd=no
ripngd=no
```

Listing 5.16: Beispiel für `/etc/quagga/daemons` um nur `zebra` zu starten

Zusätzlich benötigt jeder Daemon seine eigene Konfigurations-Datei im Verzeichnis `/etc/quagga/` mit seinem Namen und der Erweiterung `.conf`. Die zu `zebra` gehörende Konfiguration befindet sich also in der Datei `/etc/quagga/zebra.conf`.

Um weitere Einstellungen per Kommando-Zeilen-Interface (CLI) mit Hilfe von `telnet` vornehmen zu können, sollte eine minimale Konfigurations-Datei die in Listing 5.17 aufgeführten Befehle enthalten. Dazu gehört die Definition eines Passwortes (Zeile 2). Zur besseren Übersichtlichkeit empfiehlt es sich, jedem Daemon einen entsprechenden Hostnamen zu vergeben, der am Prompt des CLIs ausgegeben wird. Da eine zentrale Haltung der

Logfiles gefordert ist, sollte das Logging per `syslog`, wie in Zeile 3 angegeben, aktiviert werden. Eine solche Konfiguration bildet auch die Ausgangsbasis für die anderen Daemons.

```
hostname zebra@router
password zebra
log syslog
```

Listing 5.17: Minimal-Konfiguration von `zebra`

Nachdem diese Einstellungen vorgenommen wurden, können die Routing-Daemons wie gewünscht gestartet werden. Zu diesem Zweck kann dem Befehl

```
/etc/init.d/quagga start
```

auch ein Prioritäts-Wert übergeben werden. Dieser bestimmt bis zu welcher eingestellten Priorität die Daemons gestartet werden. Bei einem System-Start werden automatisch alle ausgewählten Routing-Prozesse gestartet.

5.2.2 Bedienung und Konfiguration von Quagga

Die Aufteilung der Routing-Aufgaben auf mehrere Daemons sorgt dafür, dass auch die Konfiguration auf mehrere Programme verteilt ist. Beim Umstieg von einem integrierten Router kann diese Eigenart gewöhnungsbedürftig erscheinen.

Beim Start eines Daemons wird seine gespeicherte Konfiguration aus einer Datei im Verzeichnis `/etc/quagga/` gelesen. Eine manuelle Änderung dieser Datei ist zu vermeiden, da im Falle eines Syntax-Fehlers der Daemon nicht startet. Stattdessen ist die Konfiguration mit Hilfe des eingebauten Kommandozeilen-Interfaces durchzuführen. Veränderungen die auf diese Weise vorgenommen werden, wirken sich automatisch auf den laufenden Betrieb aus. Dadurch ist kein Neustart erforderlich. Das bedeutet im Gegenzug, dass Änderungen explizit abgespeichert werden müssen, um auch nach einem Neustart gültig zu sein.

Zum Zugriff auf das Kommandozeilen-Interface stehen zwei Wege zur Verfügung. Einerseits hat jeder Daemon eine per Telnet zugängliche Konsole, die nur die für ihn relevanten Schlüsselwörter unterstützt. Andererseits steht mit `vtysh` auch ein Programm zur Verfügung, mit dem alle Daemons zeitgleich bedient werden können. Dieses Programm kann auch als Login-Shell genutzt werden und ermöglicht somit eine gute Annäherung an das Verhalten eines Cisco-Routers, bietet aber keinerlei Authentifizierung des Nutzers.

Um mittels Telnet auf den Haupt-Prozess `zebra` zuzugreifen, wird eine Verbindung zum zugehörigen VTY-Port (*Virtual Terminal*) hergestellt. Standardmäßig ist dieser auf 2601 festgelegt. Um diesen Vorgang zu erleichtern, findet in der Datei `/etc/services` eine Zuordnung zwischen den numerischen Ports und einem Namen statt. Dieser Name entspricht in diesem Fall dem Namen des Daemons. Somit lässt sich auf den Prozess `zebra` im lokalen System mittels

```
telnet localhost zebra
```

zugreifen.

Nach der Eingabe des definierten Passworts gelangt man in den unprivilegierten Modus, der an einem > am Ende des Prompts zu erkennen ist. In diesem Modus können lediglich verschiedene Status angezeigt werden. Um in den Betrieb des Daemons einzugreifen, ist ein Wechsel in den privilegierten Modus durch Eingabe von `enable` nötig. Mit Hilfe von `disable` kann dieser Modus wieder verlassen werden.

Bei der Eingabe von Befehlen ist die Verwendung von Abkürzungen der Schlüsselwörter möglich, solange sie eindeutig sind (`enable` und `en` sind somit identisch). Zusätzlich kann ein Schlüsselwort mittels der Tabulator-Taste vervollständigt werden. Ist die bisherige Eingabe nicht eindeutig, erscheint eine Auswahl der Möglichkeiten, etwa

```
zebra@router> e<TAB>
enable     exit
```

beim Versuch, die Abkürzung `e` zu vervollständigen. Nach einem Druck auf `?` erscheint eine kontext-sensitive Hilfe zu den derzeit möglichen Schlüsselworten. Wird diese z. B. nach der obigen Eingabe aufgerufen, erscheint folgendes Ergebnis:

```
zebra@router> e?
  enable  Turn on privileged mode command
  exit    Exit current mode and down to previous mode
```

Die eigentlichen Einstellungen werden im Konfigurations-Modus vorgenommen. Dieser wird durch Eingabe von

```
configure terminal
```

aus dem privilegierten Modus erreicht. Die in den folgenden Abschnitten dargestellten Konfigurations-Abschnitte können in dieser Reihenfolge am Prompt eingegeben werden.

Mit dem Schlüsselwort `no` vor einem Befehl kann die entsprechende Konfigurations-Option generell rückgängig gemacht werden. Der Befehl

```
no log file
```

deaktiviert z. B. ein vorher definiertes Logging in eine Datei wieder.

Nachdem die Einstellungen getestet wurden, müssen sie abgespeichert werden, so dass sie nach einem Neustart automatisch wieder geladen werden. Diesem Zweck dienen die beiden äquivalenten Befehle

```
write file
copy running-config startup-config
```

Während des Konfigurierens ist es oft hilfreich, sich die aktuelle Konfiguration vor Augen zu führen. Mit Hilfe des Befehls

```
show running-config
```

ist dies leicht möglich.

5.2.3 Grundkonfiguration

Neben den Einstellungen aus Listing 5.17, die zum Start eines Quagga-Daemons unerlässlich sind, sollten einige Optionen gesetzt werden, die für den weiteren Betrieb sinnvoll sind.

Sollen auch unprivilegierte Benutzer die Möglichkeit haben, den Status des Routers abzufragen, ohne in den Betrieb eingreifen zu dürfen, ist eine zusätzliche Passwort-Abfrage für die Benutzung des `enable`-Kommandos zu aktivieren. Der Befehl

```
enable password PASSWORT
```

setzt dieses Passwort auf den Wert „PASSWORT“ und aktiviert die Passwort-Abfrage. Derzeit hat diese Option allerdings keinen Einfluss auf die Benutzung von `vtysh`. Deshalb darf dieses Programm nur vertrauenswürdigen Nutzern zur Verfügung gestellt werden.

Standardmäßig speichert Quagga Passwörter unverschlüsselt in seinen Konfigurations-Dateien ab. Um das einfache Ausspionieren der definierten Passwörter zu vermeiden, existiert eine Option zur Ablage verschlüsselter Passwörter. Nach deren Aktivierung mit dem Befehl

```
service password-encryption
```

wird aus jedem bestehenden Passwort ein crypt-Hash erzeugt und zur Authentifizierung benutzt.

Kommentare sorgen bei vielen gleichartigen Einstellungen häufig für eine bessere Lesbarkeit der Konfiguration. Diesem Zweck dienen z. B. Interface-Beschreibungen, die eine einfachere Identifizierung ermöglichen. Die Befehle

```
interface eth0.15
 description Peering mit ISP A
```

geben dem Interface `eth0.15` die Information mit, dass es dem Peering mit *ISP A* dient.

5.2.4 IPv6 Router-Advertisements

Zu den Basis-Funktionen von IPv6 gehört für einen Host das automatische Auffinden eines Default-Gateways, um mit der Außenwelt kommunizieren zu können. Zur Unterstützung dieser in RFC2461 [NNS98] definierten *Router Discovery* (RA) ist es unerlässlich auf dem Router einen entsprechenden Daemon einzurichten.

Neben spezialisierten Programmen für diesen Zweck, wie `radvd`, bietet auch der zum Einsatz kommende Routing-Daemon Quagga eine entsprechende Funktionalität. Diese muss pro Interface konfiguriert und aktiviert werden und deckt nur die grundlegenden Möglichkeiten ab.

Listing 5.18 zeigt dies am Beispiel des Interfaces `eth0.10`. Nach dem Konfigurieren der IPv6-Adresse wird in Zeile 3 das Aussenden von RA-Paketen aktiviert. Damit werden etwaige Hosts lediglich über einen verfügbaren Default-Router informiert. Zusätzlich kann auch das für die Adress-Auto-Konfiguration notwendige IPv6-Präfix übertragen werden. Ein solches Präfix besitzt immer eine maximale und eine bevorzugte Gültigkeits-Dauer. Diese beträgt hier 1 Woche (604800 s) bzw. 2 Tage (172800 s). Zusätzlich legt `autoconfig` fest, dass das angegeben Präfix für die automatische Konfiguration benutzt werden darf.


```
interface eth0.10
 ipv6 address 2001:db8:100:10::10/64
 ipv6 nd send-ra
 ipv6 nd prefix-advertisement 2001:db8:100:10::/64 604800 172800  ↵
    onlink autoconfig
```

Listing 5.18: Konfiguration von Router-Advertisements

5.2.5 Einbinden in das IGP RIPng

Der Einsatz des einfachen Distance-Vector-Routing-Protokolls RIPng erfordert nur wenig Aufwand. Zum Start des zugehörigen Daemons `ripngd` ist eine Minimal-Konfiguration nötig, wie in Listing 5.17. Zusätzlich muss der zugehörige Eintrag in der Datei `/etc/quagga/daemons` so modifiziert werden, dass seine Start-Priorität größer als die von `zebra` verwendet wird. Die Konfiguration erfolgt daraufhin, wie in Abschnitt 5.2.3 beschrieben, bevorzugt per Telnet auf dem Port 2603 bzw. `ripngd`.

Um lediglich Routen von benachbarten Routern auf dem Interface `eth0.1` zu empfangen, genügt die folgende Konfiguration:

```
router ripng
 passive-interface eth0.1
```

Sollen dagegen auch die eigenen angeschlossenen Netze angeboten werden, wird `passive-interface` durch das Schlüsselwort `network` ersetzt. Ab diesem Moment ist der Router aktiv und sendet Angebote an seine Nachbarn an den angegebenen Schnittstellen. Diese Nachrichten umfassen aber nur die angeschlossenen Netze, deren Interface per `network` definiert oder per RIPng empfangen wurden.

Zum Anbieten von Routen, die nicht per RIPng gelernt wurden, dient das Schlüsselwort `redistribute`, mit dem generell Routen anderer Quellen in ein Routing-Protokoll übernommen werden können. Meist ist es sinnvoll, zumindest die direkt angebundenen Netze mit in die Routing-Informationen aufzunehmen. Mittels

```
router ripng
 redistribute connected
```

werden die Netze der in `zebra` konfigurierten Interfaces mit in die RIPng-Tabelle aufgenommen. Auf gleiche Weise können die dort definierten statischen Routen (`static`) und die außerhalb von Quagga erzeugten Routen, die vom Kernel übernommen wurden (`kernel`), eingefügt werden.

Bei diesem Prozess kann es vorkommen, dass auch unerwünschte Routen übernommen werden. Um dies zu verhindern, kann mittels Präfix-Listen ein Filter erstellt werden. In Listing 5.19 wird in den Zeilen 10 bis 11 eine Präfix-Liste definiert, die keine Multicast-Adresse hindurch lässt, während alle anderen Adressen erlaubt sind. Zur Aktivierung dieses Filters für ausgehende Routing-Updates dient das Schlüsselwort `distribute-list` in Zeile 8. Auf Wunsch können auch eingehende Angebote entsprechend gefiltert werden.

Zeile 7 des Beispiels zeigt, wie mehrere Routen zusammengefasst (aggregiert) werden können, um die Anzahl der angebotenen Routen möglichst gering zu halten. Dieses Vor-

gehen ist allerdings nur dann sinnvoll, wenn zum Erreichen der betreffenden Netze dieser Router durchlaufen werden muss.

```
router ripng
 network eth0.1
 network eth0.2
 redistribute connected
 redistribute static
 redistribute kernel
 aggregate-address 2001:db8:1000::/48
 distribute-list prefix only_unicast out
!
ipv6 prefix-list only_unicast seq 5 deny ff00::/8 le 128
ipv6 prefix-list only_unicast seq 10 permit any
```

Listing 5.19: Beispiel-Konfiguration für RIPng

5.2.6 Einbinden in das IGP OSPFv3

Statt eines Distance-Vector-Routing-Protokolls wie RIPng kommt in ISP-Netzwerken häufig das Link-State-Protokoll OSPF zum Einsatz, da dieses wesentlich schneller konvergiert und somit für komplexere Netz-Topologien besser geeignet ist. Quagga implementiert mit dem Daemon `ospf6d` die für OSPFv3 notwendigen Basisfunktionalitäten.

Die Aktivierung verläuft analog zu der von `ripngd` durch Modifikation der Datei `/etc/quagga/daemons` und Anlegen einer kleinen Minimal-Konfiguration. Ähnlich einfach wie bei RIPng gestaltet sich die Konfiguration, wenn der Router nur angebotene Routing-Informationen empfangen und diese erlernten Informationen weitergeben soll. Die Befehle

```
interface eth0.1
!
router ospf6
 router-id 192.168.8.10
 interface eth0.1 area 0.0.0.0
```

genügen, um mit den benachbarten Routern auf dem Interface `eth0.1` Verbindung aufzunehmen.

In diesem Beispiel ist zu erkennen, dass IPv4-Adressen für die Konfiguration verwendet werden. Dieser Umstand erklärt sich aus der Tatsache, dass OSPFv3 eine minimale Anpassung von OSPFv2 an die Bedürfnisse von IPv6 darstellt. Da in diesen Protokollen 32-Bit-Werte zur Identifizierung der Area und der Router verwendet werden, hat sich die Notatation als IPv4-Adresse durchgesetzt.

OSPF definiert zwar die Möglichkeit in einem Netzwerk verschiedene Areas zu verwenden, um die zu verteilenden Link-State-Announcements zu reduzieren. Derzeit unterstützt Quaggas OSPFv3-Implementation laut Dokumentation aber nur Area 0, den Backbone.

Wie in der Konfiguration von RIPng können Routing-Informationen aus anderen Quellen mittels entsprechender `redistribute`-Befehle in das Netz der OSPFv3-Router eingespeist werden. Das Vorgehen beim Filtern von Routen unterscheidet sich hingegen. Aktuell können nur die ausgesendeten Link-State-Advertisements gefiltert werden. Dies erfolgt allerdings für jedes Interface unabhängig, so dass eine wesentlich feinere Kontrolle möglich ist.

Mit Listing 5.20 wird eine Konfiguration erzeugt, die der obigen für RIPng ähnelt. Auf der Schnittstelle `eth0.2` werden allerdings nur Routen unterhalb von `2001:db8:1000::/40` angeboten, während den Routern an `eth0.1` die kompletten Routing-Informationen zur Verfügung gestellt werden.

Des Weiteren arbeitet OSPF mit einem feiner abgestuften System von Metriken für die Routing-Entscheidung. Dazu wird wie in Zeile 4 gezeigt einem Interface ein Kosten-Wert zugewiesen. Bei der Berechnung der Routing-Tabelle aus der Link-State-Datenbank wird zum Erreichen eines bestimmten Zieles stets der Weg mit den geringsten Kosten genutzt.

```
interface eth0.1
!
interface eth0.2
 ipv6 ospf6 cost 10
 ipv6 ospf6 advertise prefix-list kundennetze
!
router ospf6
 router-id 192.168.8.10
 redistribute connected
 redistribute static
 redistribute kernel
 interface eth0.1 area 0.0.0.0
 interface eth0.2 area 0.0.0.0
!
ipv6 prefix-list kundennetze seq 5 permit 2001:db8:1000::/40 le 128
```

Listing 5.20: Beispiel-Konfiguration für OSPFv3

Normalerweise wird in einem Netzwerk nur ein IGP eingesetzt, um den administrativen Aufwand auf ein sinnvolles Maß zu reduzieren. Es kann aber notwendig sein, mehrere IGPs parallel zu betreiben. Bei der Umstellung von einem Routing-Protokoll auf ein anderes bietet es sich an, diese wie „ships in the dark“ zu behandeln und keinen Informationsaustausch zwischen ihnen vorzunehmen. Soll dagegen z. B. ein Kunden-Netzwerk über mehrere Wege per RIPng angeschlossen werden, während intern OSPFv3 genutzt wird, müssen die Daten aus RIPng mittels `redistribute ripng` nach OSPFv3 importiert werden.

5.2.7 Außenanbindung per BGP

Im Gegensatz zu den IGPs basiert das *Border Gateway Protocol* (BGP) nicht auf der automatischen Erkennung seiner Nachbarn per Multicast. Stattdessen muss zu jedem gewünschten Partner-Router eine TCP-Verbindung aufgebaut werden, über die die Routing-Updates

ausgetauscht werden. Die BGP-Implementation von Quagga ist bereits sehr ausgereift, da es eine der am häufigsten benötigte Funktion der Routing-Suite ist.

Der Daemon `bgpd` wird wie die anderen aktiviert. Da ein funktionierendes IGP Voraussetzung für evtl. benötigte iBGP-Verbindungen ist, sollte der Wert der Start-Priorität höher gewählt werden als für die IGPs. Wenn der `bgpd` gestartet ist, kann er mit Hilfe des CLIs konfiguriert werden.

Um eine Verbindung zum Nachbar-Router `2001:db8:100:15::14` im AS65002 aufzubauen, genügt es die Befehle aus Listing 5.21 einzugeben. Mit diesem Beispiel werden sowohl IPv4- als auch IPv6-Routing-Informationen empfangen, da die Nutzung von IPv4 nicht explizit aktiviert werden braucht. Dagegen ist gut zu erkennen, dass die IPv6-Funktionalität als Multi-Protokoll-Erweiterung erst später zu BGP4 hinzugefügt wurde. Einerseits wird die 32 Bit lange Router-ID als IPv4-Adresse notiert. Andererseits bedarf es einer expliziten Auswahl der Adress-Familie IPv6 (siehe Zeile 5), um die Fähigkeit zu aktivieren IPv6-Routen behandeln zu können.

```
router bgp 65001
 bgp router-id 192.168.8.13
 neighbor 2001:db8:100:15::14 remote-as 65002
 !
 address-family ipv6
 neighbor 2001:db8:100:15::14 activate
 exit-address-family
```

Listing 5.21: Minimale Konfiguration für ein BGP-Peering

Neben dieser Basis-Konfiguration, die eine eBGP-Verbindung zwischen dem AS65001 und dem AS65002 initiiert, ist es häufig auch nötig iBGP-Verbindungen zwischen den Border-Routern eines autonomen Systems herzustellen. Dazu wird bei der Definition des Nachbars mittels

```
router bgp 65001
 neighbor 2001:db8:100:4::12 remote-as 65001
```

das selbe AS benutzt.

Mit den Befehlen

```
router bgp 65001
 address-family ipv6
 network 2001:db8::/35
```

wird dem BGP-Daemon mitgeteilt, dass er das lokale Präfix `2001:db8::/35` seinen Nachbarn per BGP anbieten soll. Um bei Verwendung von Default-Routen unnötige Routing-Schleifen zu vermeiden, empfiehlt es sich in `zebra` gleichzeitig mittels

```
ipv6 route 2001:db8::/35 ::1 reject
```

eine statische Route zum Zurückweisen von Paketen für unbekannte Ziele anzulegen. Da in der globalen Routing-Tabelle der DFZ nicht beliebig spezielle Routing-Informationen verteilt werden können, dürfen die Daten der IGPs i. d. R. nicht nach BGP importiert werden.

Normalerweise werden alle von einem AS empfangenen Updates an alle anderen verbundenen AS weitergeleitet. Dies bedeutet, dass IP-Pakete zwischen diesen fremdem AS über das eigene Netzwerk transferiert werden könnten. Da dieser Transit-Verkehr meist unerwünscht ist, sollten die angebotenen Informationen zur Erreichbarkeit gefiltert werden.

Zu diesem Zweck kann pro Nachbar eine Filter-Liste definiert werden, die dann mittels der Access-Liste

```
ip as-path access-list keinTransit permit ^$
```

nur solche Updates auswählt, die noch keinen Eintrag im Attribut `AS_PATH` enthalten. Der reguläre Ausdruck dieses Filters kann den Bedürfnissen entsprechend erweitert werden, wenn bestimmten AS Transit-Verkehr möglich sein soll.

Neben ausgehenden Routing-Updates, sind auch eingehende Updates so zu filtern, dass die Stabilität des Routings im Internet gewährleistet ist. In [Dör04] werden Empfehlungen gegeben, welche Präfixe im Internet als gültig erachtet und angenommen werden sollten. Zu den ungültigen Adress-Bereichen gehört auch das in dieser Arbeit verwendete Dokumentation-Präfix `2001:db8::/32`.

Da beim Filtern von eingehenden Routing-Updates normalerweise alle unerwünschten Einträge verworfen werden, ist es für eine Änderung der Filter-Regeln erforderlich die BGP-Verbindung ab- und wieder neu aufzubauen. Um dieses Problem zu umgehen, bietet Quagga die Möglichkeit, die ursprünglich empfangenen Updates zusätzlich zu speichern und eine sanfte Rekonfiguration durchzuführen. Auch diese Einstellung ist in die Konfiguration in Listing 5.22 eingeflossenen. Darin werden auch die übrigen Vorschläge umgesetzt.

```
router bgp 65001
 bgp router-id 192.168.8.13
 neighbor 2001:db8:100:15::14 remote-as 65002
 !
 address-family ipv6
 network 2001:db8::/35
 neighbor 2001:db8:100:15::14 activate
 neighbor 2001:db8:100:15::14 soft-reconfiguration inbound
 neighbor 2001:db8:100:15::14 prefix-list ipv6-ebgp-relaxed in
 neighbor 2001:db8:100:15::14 filter-list keinTransit out
 exit-address-family
!
ipv6 prefix-list ipv6-ebgp-relaxed permit 3ffe::/18 ge 24 le 48
ipv6 prefix-list ipv6-ebgp-relaxed permit 3ffe:4000::/18 ge 32 le 48
ipv6 prefix-list ipv6-ebgp-relaxed permit 3ffe:8000::/22 ge 28 le 48
ipv6 prefix-list ipv6-ebgp-relaxed deny 2001:db8::/32 le 128
ipv6 prefix-list ipv6-ebgp-relaxed permit 2001::/16 ge 19 le 48
ipv6 prefix-list ipv6-ebgp-relaxed permit 2002::/16
ipv6 prefix-list ipv6-ebgp-relaxed deny ::/0 le 128
!
ip as-path access-list keinTransit permit ^$
```

Listing 5.22: BGP-Konfiguration mit Filtern

5.2.8 Absicherung durch IPSec

Da Router wichtige Infrastruktur-Systeme sind, ist ein wirksamer Schutz vor unerwünschter Beeinflussung von außen angebracht. Eines der kritischsten Angriffs-Ziele stellen die BGP-Verbindungen dar, da im Falle eines Verbindungs-Abbruchs große Teile des Internets unerreichbar werden. Wie im April 2004 festgestellt wurde, kann ein außenstehender Angreifer mit relativ geringem Aufwand eine TCP-Session unterbrechen. [CER04] Zur Lösung des Problems wird die Nutzung von MD5-authentifizierten TCP-Sessions oder von IPSec angeregt. Für TCP/IPv6-Verbindungen kommt nur IPSec in Frage, da dies von allen standardkonformen IPv6-Systemen unterstützt werden muss.

Neben der IPSec-Unterstützung im Linux-Kernel werden die IPSec-Tools für Linux benötigt, um Einstellungen vornehmen zu können. Zu diesem Paket gehört ebenso der IKE[1]-Daemon `racoon`. Bei der Installation der Debian-Pakete `ipsec-tools` und `racoon` wird das Programm `racoon-tool` mitgeliefert, das die gesamte IPSec-Konfiguration in der Datei `/etc/racoon/racoon-tool.conf` zusammenfasst. Alternativ kann die Konfiguration auch manuell mit Hilfe von Skripten durchgeführt werden.

Wenn bei der Installation nicht angegeben wurde, dass `racoon-tool` benutzt werden soll, kann dies durch Änderung der Zeile

```
CONFIG_MODE="racoon-tool"
```

in der Datei `/etc/default/racoon` nachgeholt werden. Zusätzlich ist darauf zu achten, dass die Paket-Filter-Einstellungen sowohl die IPSec-Protokolle als auch IKE zulassen.

In Listing 5.23 ist eine Konfiguration dargestellt, mit deren Hilfe alle TCP-Verbindungen zum System `2001:db8:100:15::14` gegen Daten-Veränderung geschützt werden. Um den Transport-Mode mit AH (Authentication Header) benutzen zu können, muss eine spezielle Regel definiert werden. Da `racoon-tool` normalerweise die IPv6-Unterstützung von `racoon` deaktiviert, müssen auch die Aufruf-Parameter angepasst werden (siehe Zeile 3).

Das IKE-Protokoll bietet zwar die Möglichkeit mit asymmetrischen Verschlüsselungs-Verfahren zu arbeiten. Dies verursacht aber insbesondere für kleine Installationen einen erhöhten Aufwand. Deshalb wird in diesem Beispiel ein vereinbartes Passwort (PSK) verwendet, das in der Datei `/etc/racoon/psk.txt` wie folgt definiert wird:

```
2001:db8:100:15::14     Sehr_sicheres_Passwort
```

Nach einem Neustart oder dem Kommando

```
/etc/init.d/racoon restart
```

werden die so definierten Regeln aktiv. Beim ersten Verbindungsversuch, der diesen Regeln entspricht, vereinbaren die IKE-Daemons einen zeitlich beschränkten Schlüssel. Auf Grund der Implementation der IPSec-Funktionen im Linux-Kernel wird dabei das erste Paket mit dem Fehler „Destination unreachable“ quittiert, was einen neuerlichen Verbindungsversuch erforderlich macht. Im Fall von BGP wird damit der Aufbau der Routing-Verbindung geringfügig verzögert.

[1] Internet Key Exchange – Protokoll zur Vereinbarung von symmetrischen Schlüsseln


```
1  global:
2          log: notify
3          racoon_command: /usr/sbin/racoon -f ___path_racoon_conf___
4
5  connection(routerE):
6          src_ip: 2001:db8:100:15::13
7          dst_ip: 2001:db8:100:15::14
8          upperspec: tcp
9          admin_status: yes
10         spdadd_template: transportAH
11
12 spdadd(transportAH): spdadd ___src_ip___ ___dst_ip___ ___upperspec___ ↵
       -P out ipsec ah/transport//require; spdadd ___dst_ip___  ↵
     ___src_ip___ ___upperspec___ -P in ipsec ah/transport//require;
```

Listing 5.23: IPSec-Konfiguration mittels `racoon-tool`

5.3 Integration weiterer Dienste

Nachdem der Router seine Basis-Funktionalität erhalten hat, können weitere Funktionen hinzugefügt werden. Insbesondere für kleinere ISP ist es wichtig, dass ein Router möglichst viele Anwendungsfälle abdecken kann. Im Folgenden soll dem Router ermöglicht werden, L2TP-Sessions zu terminieren, sowie Multicast-Routing und MPLS zur Integration in einen MPLS-Backbone zu unterstützen.

5.3.1 L2TP zur Einwahl

Zur Anbindung von ADSL-Kunden der T-Com, bietet dieser Carrier die Übergabe per *Layer Two Tunneling Protocol* (L2TP) an. Diese Tunnel müssen an einem Router des ISPs terminiert werden. Das Debian-Paket `l2tpd` enthält die dafür notwendige Funktionalität. Da L2TP zum Transport von PPP-Datenströmen dient, sind zusätzlich die Kernel-Unterstützung für PPP und das Paket `ppp` notwendig.

Wenn diese Voraussetzungen geschaffen wurden, muss der `l2tpd` den Bedürfnissen entsprechend konfiguriert werden. Dazu sollte die Datei `/etc/l2tpd/l2tpd.conf` dem in Listing 5.24 aufgeführten Beispiel ähneln. Diese Konfiguration besteht aus mehreren Sektionen und bietet somit die Möglichkeit unterschiedlicher Einstellungen für verschiedene Anwendungsfälle oder für diverse Carrier.

Die in der Sektion `[lns default]` aufgeführten Einstellungen führen zu einer entsprechenden Parametrisierung des `pppd`, der beim Aufbau einer neuen Verbindung gestartet wird. Diese L2TP-Implementation erfordert zwingend eine dynamische Zuweisung der IPv4-Adresse aus einem Pool, wie er in Zeile 9 definiert ist. Die Zuweisung einer für einen bestimmten Benutzer bestimmten IPv4-Adresse ist dem `pppd` mit Hilfe eines RADIUS-Plugins möglich. Die Optionen, die hier angegeben werden können, sind i. d. R. nicht ausreichend, um eine PPP-Verbindung aufbauen zu können. Deshalb wird in Zeile 17 eine Datei mit zusätzlichen Optionen für den `pppd` definiert.

Listing 5.25 stellt den Inhalt einer solchen Options-Datei dar. Vor allen Dingen ist es

wichtig, dem PPP-Daemon mitzuteilen, dass für die Verbindung kein Modem eingesetzt wird. In den per L2TP übergebenen PPP-Sessions werden vom LAC bereits einige Parameter ausgehandelt. Die Standard-Einstellungen des `pppd` sind dazu aber inkompatibel, so dass die Zeilen 7 bis 9 notwendig sind, damit überhaupt Daten übertragen werden können. Die Möglichkeit die zu verwendenden DNS-Server per PPP zu übermitteln, so dass der Client diesen automatisch konfigurieren kann, sollte i. d. R. genutzt werden. Die Unterstützung von IPv6 muss auf beiden Seiten aktiviert werden, damit entsprechende Aushandlungen stattfinden können. In Zeile 19 werden die für die Bildung der link-lokalen Adresse verwendeten Interface-IDs definiert.

Für die gewünschte Fähigkeit Benutzer per RADIUS zu authentifizieren sind neben den beiden letzten Zeilen in Listing 5.25 weitere Schritte notwendig. Im Quelltext des PPP-Daemons ist zwar die RADIUS-Unterstützung enthalten, aber normalerweise nicht aktiviert. Zur Aktivierung ist es sinnvoll, den Source-Code des Debian-Paketes als Ausgangsbasis zu nehmen, die notwendige Änderung vorzunehmen, das Paket zu compilieren und auf dem Router mittels `dpkg` zu installieren.

Der Quelltext für das Debian-Paket kann mittels

```
apt-get source ppp
```

bezogen werden, wenn in der Datei `/etc/apt/sources.list` eine entsprechende `deb-src`-Zeile enthalten ist. Auch wenn die notwendige Änderung durch Auskommentieren einer Zeile erfolgen könnte, ist es für das Debian-Source-Paket notwendig, eine Patch-Datei (siehe Listing 5.26) zu erzeugen und im Verzeichnis `debian/patches` abzulegen. Ist dies erfolgt, kann mit dem Befehl

```
dpkg-buildpackage
```

das Binär-Paket erzeugt werden.

Ist die so veränderte Version des `ppp`-Pakets auf dem Router installiert, muss das RADIUS-Plugin konfiguriert werden. Diese Einstellung umfasst die RADIUS-Server, die zur Authentifizierung und zum Accounting verwendet werden sollen. Diesem Zweck dienen die Einstellungen in der Datei `/etc/radiusclient/radiusclient.conf`, die in Listing 5.27 beispielhaft dargestellt sind. Für die Verbindung mit den RADIUS-Servern wird ein Passwort verwendet, das in der Datei `/etc/radiusclient/servers` spezifiziert wird.

Beim Aufbau der PPP-Verbindung wird den beiden Gegenseiten nur eine link-lokale IPv6-Adresse zugewiesen. Daher müssen in einem weiteren Schritt globale IPv6-Adressen hinzugefügt werden. Dazu können sowohl Router-Advertisements, als auch DHCPv6 eingesetzt werden. Die dafür notwendigen Informationen über die zu verwendenden IPv6-Präfixe werden auf RADIUS-Server in den Attributen `Framed-IPv6-Prefix` und `Framed-IPv6-Route` verwaltet.

Das PPP-Plugin `radattr` ist in der Lage alle vom RADIUS-Server übermittelten Attribute in einer Datei nach dem Muster `/var/run/radattr.<interface>` abzulegen. Diese Daten können dann z. B. mit den Start-Skripten des `pppd` ausgewertet werden. Da die `radiusclient`-Bibliothek derzeit keine IPv6-Datentypen unterstützt, werden diese nicht abgespeichert. Somit ist die automatische Vergabe globaler IPv6-Adressen per RADIUS derzeit nicht möglich.


```
[global]                              ; Global parameters:
port = 1701                           ; * Bind to port 1701
auth file = /etc/l2tp/l2tp-secrets    ; * challenge secrets
access control = no                   ; * No check for source
rand source = dev                     ; Source for entropy for random
;
[lns default]                         ; LNS definition
exclusive = no                        ; * Allow >1 tunnels per LAC
ip range = 10.0.100.1-10.0.100.254    ; * Allocate from this IP range
local ip = 192.168.8.10               ; * Our local IP to use
require chap = no                     ; * Require CHAP auth. by peer
refuse pap = no                       ; * Refuse PAP authentication
refuse chap = no                      ; * Refuse CHAP authentication
require authentication = yes          ; * Require peer to authenticate
name = router                         ; * Report this as our hostname
ppp debug = no                        ; * Turn on PPP debugging
pppoptfile = /etc/ppp/options.l2tpd.lns ; * ppp options file
```

Listing 5.24: Beispiel für `/etc/l2tpd/l2tpd.conf`

```
# DNS-Server im Client konfigurieren
ms-dns 192.168.8.2
# Authentifizierung erforderlich
auth
# kein Modem, sondern virtuelles Interface
nocrtscts
noaccomp
default-asyncmap
nopcomp
# MTU/MRU fuer T-DSL mit Transport ueber L2TP
mru 1456
mtu 1456
# Pings
lcp-echo-interval 30
lcp-echo-failure 4
# Nur IP transportieren
noipx
# Initiiere IPv6-Vereinbarungen
ipv6 ::1,::2
# RADIUS verwenden
plugin radius.so
plugin radattr.so
```

Listing 5.25: PPP-Optionen für L2TP-Tunnel – `/etc/ppp/options.l2tpd.lns`

```
--- ppp.old/pppd/plugins/Makefile.linux  ↵
   2004-04-27 15:37:09.000000000 +0200
+++ ppp/pppd/plugins/Makefile.linux      ↵
   2004-04-27 15:39:29.000000000 +0200
@@ -6,7 +6,7 @@

 SUBDIRS := rp-pppoe
 # Uncomment the next line to include the radius authentication plugin
-# SUBDIRS += radius
+SUBDIRS += radius
 PLUGINS := minconn.so passprompt.so passwordfd.so pppoatm.so

 # include dependencies if present
```

Listing 5.26: Patch zur Aktivierung von RADIUS in `ppp`

```
authserver radius1.firma.bsp:1812
authserver radius2.firma.bsp:1812
acctserver radius1.firma.bsp:1813
acctserver radius2.firma.bsp:1813
```

Listing 5.27: Definition der RADIUS-Server

5.3.2 MPLS

Die aktuellste Implementation von *Multiprotocol Label Switching* (MPLS) für Linux wird von James Leu entwickelt[2] und stellt neben der Forwarding-Engine im Kernel auch Programme zur Administration zur Verfügung. Zusätzlich wird eine Integration des *Label Distribution Protocols* (LDP) in Quagga/Zebra angeboten. Quagga unterstützt bisher schon den Austausch von Labels per BGP, ist aber noch nicht in der Lage die Forwarding-Engine entsprechend zu konfigurieren. Auch für andere Routing-Protokolle wird an der Integration von *Traffic-Engineering*-Funktionen gearbeitet.

All diese Entwicklungen sind derzeit noch nicht in den Standard-Umfang der betreffenden Programme eingegangen. Stattdessen finden sich im Quellcode-Repository des Projektes veränderte ältere Versionen von Linux, Quagga und anderen Werkzeugen. Daraus können mittels `diff` Patches gewonnen werden, mit denen die Integration in aktuelle Versionen möglich wird. Auf diese Weise konnte ein Patch für das Linux 2.6.6 Debian-Paket erstellt werden, das auf dem beigefügten Datenträger enthalten ist.

Damit sollte es möglich sein, manuelle MPLS-Tunnel zwischen zwei Routern anzulegen. Da ohne LDP auf diese Weise keine Interoperabilität mit den vorhandenen Cisco-Routern herzustellen war, entfällt die Erklärung dieser Konfiguration.

Mit den vorhandenen Quellen war es dagegen nicht möglich, die LDP-Implementation in einen funktionstüchtigen Zustand zu überführen. Ein entsprechender Patch für das Quagga-Debian-Paket liegt dieser Arbeit bei, führt aber bei Benutzung des `mplsd` unter verschiedenen Umständen zu Abstürzen des Daemons. Da der Haupt-Entwicker des MPLS-Linux-Projektes in [Leu04] einräumt, dass aktuell noch viel Arbeit investiert werden muss, um eine lauffähige Umgebung für LDP zu schaffen, kann unter Linux derzeit kein MPLS genutzt werden.

[2] siehe `http://mpls-linux.sourceforge.net/`

5.3.3 Multicast-Routing

Zum aktuellen Zeitpunkt unterstützt der Linux-Kernel kein IPv6-Multicast-Routing. Das *USAGI-Projekt* [USA] hat es sich aber u. a. zur Aufgabe gemacht, dies zu ändern. Eine konkrete Planung existiert allerdings noch nicht.

Nachdem die Weiterleitung der Multicast-Pakete durch den Kernel ermöglicht wurde, müssen im nächsten Schritt die Multicast-Routing-Protokolle implementiert werden. Dazu gehören sowohl Daemons für *Protocol Independent Multicast* (PIM), aber auch MBGP-Unterstützung für IPv6-Multicast. Diese sollten nach Möglichkeit zu Quagga hinzugefügt werden. In Anbetracht der Tatsache, dass mit den BSD-Derivaten bereits andere freie Betriebssysteme das Forwarding unterstützen, könnte diese Arbeit bereits beginnen. Derzeit besteht aber von Seiten des Marktes kein Druck Multicast anzubieten.

5.4 Netzwerk-Management per SNMP

Um den aufgebauten Router aus der Ferne überwachen und steuern zu können, soll das *Simple Network Management Protocol* (SNMP) zum Einsatz kommen. Neben der bloßen Installation und Konfiguration des SNMP-Agents auf dem Router kann dieser so erweitert werden, dass auch Quagga per SNMP angesprochen werden kann.

5.4.1 Installation und Konfiguration des SNMP-Agents

Zur Installation des SNMP-Agents des *NET-SNMP Projekts* wird das Debian-Paket `snmpd` benötigt. Daneben können auch die im Paket `snmp` enthaltenen Client-Programme installiert werden, um einen Zugriff auf die per SNMP verfügbar gemachten Daten zu ermöglichen.

Nach erfolgreicher Installation müssen neben einigen Informationen zum System, auf dem der Agent läuft, auch die Zugriffsberechtigungen konfiguriert werden. Zur Vereinfachung der Konfiguration steht das Programm `snmpconf` zur Verfügung. Da dieses nicht dem aktuellen Stand der Konfigurations-Syntax entspricht, sollte darauf verzichtet werden. Stattdessen sollten die vorgestellten Einstellungen in der Datei `/etc/snmp/snmpd.local.conf` vorgenommen werden.

Das Beispiel in Listing 5.28 zeigt, wie mit Hilfe des feinstufigen Rechte-Systems des SNMP-Agents der lesende Zugriff auf alle Daten per SNMP Version 2c mit der Community `Geheim` gestattet wird.

Zu diesem Zweck muss zuerst die Zuordnung der Community zu einem Security-Namen (hier `ROSec`) erfolgen. Dabei kann auch definiert werden, von welchen Adressen Zugriffe erlaubt werden sollen. Für die eigentliche Definition der Sicherheits-Richtlinien, wird eine Gruppe (`ROGroup`) definiert, der die Security-Namen angehören können. Gleichzeitig wird eine Ansicht (`all`) des Variablen-Baums definiert. Die Verknüpfung von Berechtigung und Ansicht erfolgt mit Hilfe des `access`-Schlüsselwortes.

Um einen Zugriff auf den SNMP-Agenten per IPv6 zu erlauben, wird im Listing 5.28 in Zeile 11 definiert, welche Adress-Familien und Ports für eingehende Verbindungen zugelassen werden sollen. Nach der Definition von Aufstellungsort und eines Ansprechpartners sind auch diese Daten abrufbar.

Der SNMP-Agent ist außerdem in der Lage, zusätzliche System-Werte zu überwachen und auf Anormalitäten zu reagieren. Dazu gehören Dateisystem-Auslastung oder laufende Prozesse. Des Weiteren werden verschiedene Schnittstellen angeboten, um weitere Parameter anderer Prozesse zur Verfügung zu stellen.

5.4.2 Einbinden von Quagga

Die Quagga-Daemons bieten die Möglichkeit, die von ihnen verwalteten Daten per SNMP verfügbar zu machen. Dazu wird die standardisierte SMUX[3]-Schnittstelle genutzt. Auf diesem Weg werden aktuell nur die IPv4-spezifischen MIBs für IP-Forwarding, RIPv2,

[3] SNMP multiplexing protocol, siehe RFC1227


```
#          name     source              community
com2sec    ROSec    192.168.8.0/24      Geheim
com2sec6   ROSec    2001:db8:10:1::/64  Geheim
#          name     sec.model  sec.name
group      ROGroup  v2c        ROSec
#          name     incl/excl  subtree  mask
view       all      included   .1       80
#      group   context sec.model sec.level match read write notif
access ROGroup ""      any       noauth    exact all  none  none

agentaddress udp:161,udpipv6:161

syslocation Rechenzentrum I, Schrank 4.2
syscontact  Nimda Beispiel <beispiel@firma.bsp>
```

Listing 5.28: Beispiel für `/etc/snmp/snmpd.local.conf`

OSPFv2 und BGP4 zur Verfügung gestellt. Sobald der Standardisierungs-Prozess für die IPv6-Routing-Protokolle abgeschlossen ist, werden die entsprechenden MIBs voraussichtlich auch von Quagga implementiert.

Um die Verbindung zwischen Quagga und dem SNMP-Agent herzustellen, muss dieser so konfiguriert werden, dass Verbindungs-Versuche von den Quagga-Daemons per *SMUX* angenommen werden. Dazu dient das Schlüsselwort `smuxpeer`, dem die zu verwendende OID übergeben wird. Pro Daemon wird, wie in Listing 5.29 gezeigt, ein solcher Eintrag benötigt. Nachdem der betreffende Quagga-Daemon eine Verbindung zum SNMP-Agent herstellen konnte, erfolgt die Registrierung zusätzlicher MIBs.

```
smuxpeer .1.3.6.1.4.1.3317.1.2.1    # zebra
smuxpeer .1.3.6.1.4.1.3317.1.2.2    # bgpd
smuxpeer .1.3.6.1.4.1.3317.1.2.3    # ripd
smuxpeer .1.3.6.1.4.1.3317.1.2.4    # ripngd
smuxpeer .1.3.6.1.4.1.3317.1.2.5    # ospfd
smuxpeer .1.3.6.1.4.1.3317.1.2.6    # ospf6d
```

Listing 5.29: Konfiguration des `snmpd` zur Anbindung von Quagga

Prinzipiell besteht die Möglichkeit, die Verbindung per SMUX-Protokoll mit einem Passwort abzusichern. Im Laufe der Versuche wurden die Verbindungsversuche vom SNMP-Agent abgewiesen, obwohl das Passwort korrekt übermittelt wurde, so dass darauf verzichtet werden musste. Da diese Kommunikation nur auf dem lokalen System abläuft und auf einem Router normalerweise keine Benutzer aktiv sind, sind die negativen Auswirkungen vernachlässigbar klein.

6 Test der geforderten Funktionalität

Um die korrekte Funktion der beschriebenen Konfigurationen zu prüfen, stand einerseits ein Router zur Verfügung, der eine Verbindung zum Produktions-Netzwerk der Logivision GmbH hatte und vor allen Dingen zum Test von BGP zum Einsatz kam. Andererseits wurde eine Test-Umgebung aufgebaut, die mittels User-Mode-Linux mehrere Router simuliert. Diese virtuellen Maschinen können z. B. auf einem Host-System mit Debian GNU/Linux und dem Paket `user-mode-linux` benutzt werden. Auf diese Weise lassen sich auch komplexe Szenarien inkl. Ethernet-Switch simulieren ohne physische Geräte in einem Test-Labor vorzuhalten.

Die so geschaffene Test-Umgebung basiert allerdings noch auf einem Linux-Kernel 2.4.24, da alle Versuche scheiterten, einen Kernel 2.6.6 im User-Mode zu betreiben. Bis auf MPLS ließen sich alle Tests hiermit durchführen. Abbildung 6.1 zeigt die verwendete Netzwerk-Topologie.

Neben den so implementierten Routern, ist auch ein Management-Server vorhanden, der gleichzeitig die Funktion als DNS-, NTP-, RADIUS- und Syslog-Server übernimmt und in den betreffenden Konfigurations-Dateien entsprechend angegeben wurde.

Diese Test-Umgebung ist auf der beigefügten CD enthalten. Anhang C geht näher auf die Nutzung der virtuellen Systeme ein.

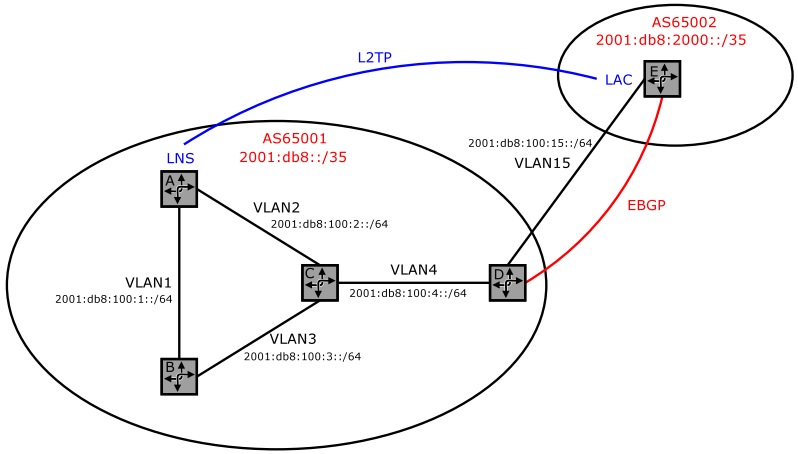

Abbildung 6.1: Topologie des Test-Netzwerks

6.1 Grundsystem

Zu testen war, ob sich der Benutzer `root` sowie der unprivilegierte Benutzer `admin` am System anmelden können. Dazu wurde an einem System im Management-Netzwerk der Befehl

```
ssh root@routerA.firma.bsp
```

verwendet und mit dem Nutzer `admin` wiederholt. Auf diese Weise wird zusätzlich geprüft, ob die Netzwerk-Verbindung besteht und die Freischaltung für Verkehr aus dem Management-Netzwerk funktionstüchtig ist.

Mit dem Befehl `ntpdc` kann der lokale NTP-Server des Routers gesteuert werden. Dass eine Verbindung zu einem anderen Server aufgebaut und die Zeit mit diesem synchronisiert wird, ist in der Ausgabe des Befehls `peers`

```
ntpdc> peers
     remote           local      st poll reach  delay   offset    disp
==============================================================================
*ntp             192.168.8.10     3   64   37 0.00104  0.000061 0.66290
```

an dem vorangestellten Stern und dem geringen Offset-Wert zu erkennen. Außerdem wurde somit die korrekte Funktion der Namens-Auflösung nachgewiesen, da der NTP-Server mit seinem Namen in der Konfiguration eingetragen war, der per DNS aufgelöst werden musste.

Auf dem System, das durch `syslog.firma.bsp` identifiziert wird, musste geprüft werden, ob die Log-Meldungen des Routers korrekt eintreffen. Dazu wurde die Datei `/var/log/syslog` betrachtet. Meldungen wie

```
Jun  2 17:19:07 routerA syslogd 1.4.1#14: restart.
```

zeigen, dass dies gegeben ist.

Die korrekte Funktion von VLANs wurde überprüft, indem auf zwei virtuellen Routern das VLAN 1 auf dem Interface `eth0` durch Eintragung in der Datei `/etc/network/interfaces` aktiviert wurde. Beim Test mittels

```
ping -s 1472 10.0.1.11
```

stellte sich heraus, dass die virtuellen Netzwerk-Schnittstellen nicht in der Lage sind IP-Pakete mit einer MTU von 1500 Bytes über ein VLAN zu transportieren. Bei Verwendung einer physischen Netzwerk-Karte mit dem `e100`-Treiber war dies möglich. In der Folge kam für VLANs in der Test-Umgebung aus diesem Grunde eine MTU von 1496 Bytes zum Einsatz.

Mit Hilfe des Parameters `-L` für `iptables` und `ip6tables` können die installierten Paket-Filter-Regeln angezeigt werden. Die Ausgabe in Listing 6.1 zeigt, dass die Regeln für IPv6 vollständig eingetragen wurden. Das zu einer Regel gehörende Interface wird hierbei nicht ausgegeben. Die korrekte Funktion des Paket-Filters wurde nicht getestet, da aufgrund positiver Erfahrungen in anderen Umfeldern von dessen weitestgehender Fehlerfreiheit ausgegangen werden kann.

```
routerA:~# ip6tables -n -L INPUT
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     esp      ::/0                 ::/0
ACCEPT     ah       ::/0                 ::/0
ACCEPT     udp      ::/0                 ::/0       udp spt:500 dpt ↵
    :500
ACCEPT     udp      2001:db8:8::/64      ::/0       udp dpt:123
ACCEPT     udp      2001:db8:8::/64      ::/0       udp spt:53
ACCEPT     tcp      ::/0                 ::/0       tcp flags:!0x16/0 ↵
    x02
ACCEPT     icmpv6   ::/0                 ::/0
ACCEPT     tcp      ::/0                 ::/0       tcp dpt:179
ACCEPT     all      fe80::/64            ::/0
ACCEPT     all      ::/0                 ::/0
ACCEPT     all      ::/0                 ::/0
DROP       all      ::/0                 ::/0
```

Listing 6.1: Ausgabe der Paket-Filter-Konfiguration

6.2 Routing-Protokolle

Um die korrekte Funktion der Routing-Protokolle zu überprüfen, genügt es, die Routing-Tabellen von Router A und Router D zu verifizieren. Dazu kann der Status jedes einzelnen Routing-Daemons abfragt werden. Die alternative Nutzung der in `zebra` zusammengeführten Informationen reduziert den Aufwand stark.

Die Listings 6.2 und 6.3 zeigen die Ausgabe des Befehls `show ipv6 route`, mit dem die IPv6-Routing-Informationen angezeigt werden können. Die Quelle jedes einzelnen Eintrags steht in der ersten Spalte. Es ist zu erkennen, dass viele Einträge doppelt vorkommen. Das ist auf den Empfang der Routen sowohl per OSPFv3 als auch RIPng zurückzuführen. Ein Stern in der dritten Spalte zeigt an, dass dieser Eintrag dem Kernel übergeben wurde und somit zum Forwarding genutzt wird. Diese Entscheidung wird anhand der in eckigen Klammern gezeigten Priorität getroffen.

Aus den vorliegenden Informationen ist zu erkennen, dass alle IPv6-Netze des ISP-Netzwerks sowohl per RIPng als auch OSPFv3 verfügbar sind. Zusätzlich wird von Router D eine Default-Route in die IGPs eingespeist, während genauere Informationen über die Erreichbarkeit des AS65002 per BGP ausgetauscht werden.

Der Ausfall eines Routers wurde ebenfalls getestet und sorgte nach dem Konvergieren der Routing-Protokolle für Nicht-Erreichbarkeit der betroffenen Teil-Netze.

```
zebra@router> show ipv6 route
Codes: K - kernel route, C - connected, S - static, R - RIPng,
       O - OSPFv3, I - ISIS, B - BGP, * - FIB route.

O>* ::/0 [110/0] via fe80::fcfd:c0ff:fea8:80c, eth0.2, 00:00:12
R   ::/0 [120/0] via fe80::fcfd:c0ff:fea8:80c, eth0.2, 00:23:38
C>* ::1/128 is directly connected, lo
O>* 2001:db8::/35 [110/0] via fe80::fcfd:c0ff:fea8:80c, eth0 ↵
```



```
    .2, 00:00:12
R   2001:db8::/35 [120/0] via fe80::fcfd:c0ff:fea8:80c, eth0 ↵
    .2, 00:23:38
O   2001:db8:10:1::/64 [110/0] via fe80::fcfd:c0ff:fea8:80b, eth0 ↵
    .1, 00:00:12
K * 2001:db8:10:1::/64 is directly connected, eth0
C>* 2001:db8:10:1::/64 is directly connected, eth0
O   2001:db8:100:1::/64 [110/0] is directly connected, eth0 ↵
    .1, 00:00:12
K * 2001:db8:100:1::/64 is directly connected, eth0.1
C>* 2001:db8:100:1::/64 is directly connected, eth0.1
O   2001:db8:100:2::/64 [110/0] is directly connected, eth0 ↵
    .2, 00:00:12
K * 2001:db8:100:2::/64 is directly connected, eth0.2
C>* 2001:db8:100:2::/64 is directly connected, eth0.2
O>* 2001:db8:100:3::/64 [110/0] via fe80::fcfd:c0ff:fea8:80c, eth0 ↵
    .2, 00:00:12
R   2001:db8:100:3::/64 [120/0] via fe80::fcfd:c0ff:fea8:80b, eth0 ↵
    .1, 00:24:06
O>* 2001:db8:100:4::/64 [110/0] via fe80::fcfd:c0ff:fea8:80c, eth0 ↵
    .2, 00:00:12
R   2001:db8:100:4::/64 [120/0] via fe80::fcfd:c0ff:fea8:80c, eth0 ↵
    .2, 00:23:38
K * 2001:db8:100:10::/64 is directly connected, eth0.10
C>* 2001:db8:100:10::/64 is directly connected, eth0.10
O>* 2001:db8:100:15::/64 [110/0] via fe80::fcfd:c0ff:fea8:80c, eth0 ↵
    .2, 00:00:12
R   2001:db8:100:15::/64 [120/0] via fe80::fcfd:c0ff:fea8:80c, eth0 ↵
    .2, 00:23:38
C * fe80::/64 is directly connected, eth0.10
C * fe80::/64 is directly connected, eth0.1
C * fe80::/64 is directly connected, eth0.2
K * fe80::/64 is directly connected, eth0.10
C>* fe80::/64 is directly connected, eth0
K>* ff00::/8 is directly connected, eth0.10
```

Listing 6.2: Router A: Ausgabe von `show ipv6 route`

```
zebra@router> show ipv6 route
Codes: K - kernel route, C - connected, S - static, R - RIPng,
       O - OSPFv3, I - ISIS, B - BGP, * - FIB route.

S>* ::/0 [1/0] is directly connected, lo, rej
C>* ::1/128 is directly connected, lo
S>* 2001:db8::/35 [1/0] is directly connected, lo, rej
O   2001:db8:10:1::/64 [110/0] via fe80::fcfd:c0ff:fea8:80c, eth0 ↵
    .4, 00:23:43
K * 2001:db8:10:1::/64 is directly connected, eth0
C>* 2001:db8:10:1::/64 is directly connected, eth0
O>* 2001:db8:100:1::/64 [110/0] via fe80::fcfd:c0ff:fea8:80c, eth0 ↵
    .4, 00:01:03
R   2001:db8:100:1::/64 [120/0] via fe80::fcfd:c0ff:fea8:80c, eth0 ↵
    .4, 00:24:28
```

```
O>* 2001:db8:100:2::/64 [110/0] via fe80::fcfd:c0ff:fea8:80c, eth0 ↲
   .4, 00:01:06
R   2001:db8:100:2::/64 [120/0] via fe80::fcfd:c0ff:fea8:80c, eth0 ↲
   .4, 00:24:28
O>* 2001:db8:100:3::/64 [110/0] via fe80::fcfd:c0ff:fea8:80c, eth0 ↲
   .4, 00:23:43
R   2001:db8:100:3::/64 [120/0] via fe80::fcfd:c0ff:fea8:80c, eth0 ↲
   .4, 00:24:28
O   2001:db8:100:4::/64 [110/0] is directly connected, eth0 ↲
   .4, 00:23:46
C>* 2001:db8:100:4::/64 is directly connected, eth0.4
O>* 2001:db8:100:10::/64 [110/0] via fe80::fcfd:c0ff:fea8:80c, eth0 ↲
   .4, 00:01:02
R   2001:db8:100:10::/64 [120/0] via fe80::fcfd:c0ff:fea8:80c, eth0 ↲
   .4, 00:24:28
C>* 2001:db8:100:15::/64 is directly connected, eth0.15
B>* 2001:db8:2000::/35 [20/0] via fe80::fcfd:c0ff:fea8:80e, eth0 ↲
   .15, 00:08:36
C * fe80::/64 is directly connected, eth0.15
C * fe80::/64 is directly connected, eth0.4
K * fe80::/64 is directly connected, eth0
C>* fe80::/64 is directly connected, eth0
O   ff00::/8 [110/0] via fe80::fcfd:c0ff:fea8:80c, eth0.4, 00:01:02
K>* ff00::/8 is directly connected, eth0
```

Listing 6.3: Router D: Ausgabe von `show ipv6 route`

6.3 Weiterleitung von Paketen

Nachdem die Routing-Tabellen aufgebaut sind, kann getestet werden, ob und mit welcher Geschwindigkeit die Pakete vom Router weitergeleitet werden. Da hierfür kein realistisches Test-System zur Verfügung stand, beschränkten sich die Tests darauf, die grundsätzliche Fähigkeit zu überprüfen.

Mit Hilfe des Befehls `traceroute6` kann der Weg zu einem angegebenen Ziel verfolgt und somit die korrekte Funktion des Routings überprüft werden. Listing 6.4 gibt den kompletten Weg von Router A zum LAN-Interface von Router E wieder. Dies zeigt, dass die Router C und D die Pakete korrekt weiterleiten.

```
routerA:~# traceroute6 -n 2001:db8:2000::1
traceroute to 2001:db8:2000::1 from 2001:db8:100:2::10, 30 hops max ↲
   , 16 byte packets
 1  2001:db8:100:2::12  2.153 ms  1.853 ms  1.965 ms
 2  2001:db8:100:4::13  4.145 ms  3.745 ms  3.775 ms
 3  2001:db8:2000::1  5.067 ms  5.558 ms  5.825 ms
```

Listing 6.4: Routing-Weg von Router A zu Router E

6.4 Aufbau eines L2TP-Tunnels

Neben der beschrieben Konfiguration des LNS, wird zum Test der korrekten Funktion ein LAC benötigt. Dieses wurde auf Router E ebenfalls mit dem `l2tpd` implementiert, der mit der IP-Adresse 10.9.8.7 arbeitet. Statt bei einem abgesetzten Einwahl-Nutzers, wird die PPP-Verbindung auf dem selben System terminiert.

Nach dem Initiieren der L2TP-Verbindung mittels

```
echo "call diplom" > /var/run/l2tp-control
```

erscheinen im Logfile die Informationen über die auf der PPP-Verbindung verwendeten IPv4- und IPv6-Adressen. Die Zeilen

```
local  IP address 192.168.8.10
remote IP address 10.0.99.1
local  LL address fe80::0000:0000:0000:0001
remote LL address fe80::0000:0000:0000:0002
```

zeigen somit an, unter welchen Adressen der Client erreichbar ist.

Zur Prüfung kommen die Programm `ping` und `ping6` zum Einsatz. Im Falle der link-lokalen IPv6-Adressen muss dafür zusätzlich das zu verwendende Interface angegeben werden. Listing 6.5 zeigt eine erfolgreiche Datenübertragung.

```
routerA:~# ping -c 1 10.0.99.1
PING 10.0.99.1 (10.0.99.1) 56(84) bytes of data.
64 bytes from 10.0.99.1: icmp_seq=1 ttl=64 time=8.39 ms
--- 10.0.99.1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 8.393/8.393/8.393/0.000 ms

routerA:~# ping6 -c 1 -I ppp0 fe80::2
PING fe80::2(fe80::2) from fe80::1 ppp0: 56 data bytes
64 bytes from fe80::2: icmp_seq=1 ttl=64 time=7.91 ms
--- fe80::2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 7.913/7.913/7.913/0.000 ms
```

Listing 6.5: L2TP-Verbindungs-Test mittels `ping`

Dieses Vorgehen dient gleichzeitig dem Nachweis der korrekten Funktion des IPv4-Forwardings, da zum Transport des L2TP-Tunnels eine IPv4-Verbindung zwischen den Routern A und E genutzt wird.

6.5 Status-Abfrage per SNMP

Zum Test der korrekten Funktion von SNMP-Abfragen, wurde der Name des Routers mittels

```
# snmpget -v 2c -c Geheim routerA sysName.0
SNMPv2-MIB::sysName.0 = STRING: routerA
```

abgefragt. Dieses Ergebnis zeigt, dass der SNMP-Agent korrekt arbeitet. Um zu prüfen, dass Quagga seine unterstützten MIBs richtigt registriert hat, wurde mir dem Befehl

```
# snmpget -v 2c -c Geheim routerA 1.3.6.1.2.1.4.24.1.0
IP-MIB::ip.24.1.0 = Gauge32: 12
```

ein Wert aus der IP-Forwarding-MIB mit dem Befehl abgefragt. Die Zahl 12 gibt die Zahl der aktuell von `zebra` verwalteten IPv4-Routing-Tabellen-Einträge an.

Von Quagga werden derzeit keine IPv6-spezifischen Daten zur Verfügung gestellt. Bevor diese Unterstützung nicht implementiert ist, lassen sich nur IPv4-betreffende Routing-Informationen abfragen. Für den Transport der SNMP-Requests kann IPv6 dagegen bereits eingesetzt werden.

7 Ergebnisse

7.1 Vergleich mit einem Cisco-Router

Da sich ein selbst entwickelter Router mit kommerziell erhältlichen Geräten messen muss, folgt nun ein Vergleich mit einem Router vom Marktführer *Cisco Systems, Inc.*. In einem ähnlichen Leistungs-Segment agieren die Router der 7200er-Serie, die bei ISPs häufig zum Einsatz kommen. Diese unterscheiden sich vor allen Dingen in der Zahl der verfügbarer Slots für Netzwerk-Interfaces, so dass keine spezielle Modellauswahl notwendig ist.

Kriterium	Cisco 72xx	Linux mit Quagga
Art der Forwarding-Engine	für IPv4 Hardware-beschleunigt, IPv6 größtenteils in der Haupt-CPU, mit Cisco-Express-Forwarding (CEF) weniger abhängig von Cache-Effekten, somit geringere Anfälligkeit für DDoS-Attacken	Forwarding komplett in Software, Routing-Tabelle mit Route-Cache, im Falle von DDoS-Attacken kann dies zu Unerreichbarkeit führen
Konformität mit IPv6-Standards	Bei Verwendung sehr aktueller IOS-Releases, sehr gut	Seit Linux-Kernel 2.6.x, sehr gut; für aktuellste Entwicklungen sind USAGI-Patches sinnvoll; alle notwendigen Services sind in guter Qualität vorhanden
Unterstützung der Routing-Protokolle RIPng, OSPFv3, BGP mit Multiprotokoll-Erweiterungen für IPv6	In aktuellen IOS-Releases, sehr gut; da diese z. T. nicht stabil im Produktionsbetrieb verwendbar sind, teilweise nicht komplett implementiert	RIPng vollständig; OSPFv3 nur Area 0, noch nicht alle Funktionen wie für IPv4 vorhanden; BGP4+ wird größtenteils unterstützt
Anzahl der terminierbaren L2TP-Tunnel	ca. 8000; empfohlen: max. 500	ca. 250, beschränkt durch maximal verfügbare Datei-Deskriptoren

Kriterium	Cisco 72xx	Linux mit Quagga
Integration eines Paket-Filters	langsam, da lineare Regel-Liste verwendet wird	schnell und flexibel wegen möglicher Baum-Struktur und schnellem Prozessor
Multicast-Routing für IPv6	sowohl IPv6-Multicast für BGP als auch PIM-SM sind in sehr aktuellen IOS-Releases implementiert; Multicast-Forwarding funktionsfähig	derzeit kein IPv6-Multicast-Forwarding im Kernel verfügbar; Quagga beinhaltet derzeit keine IPv6-Multicast-Funktionalität
Unterstützung von MPLS	MPLS wird von verschiedenen IOS-Releases in unterschiedlicher Qualität unterstützt; in aktuellen Versionen ist der Betrieb für IPv4 problemlos möglich; IPv6 nicht vollständig integriert	MPLS beschränkt sich auf eine experimentelle Forwarding-Engine; die zu unterstützenden Verfahren zum Label-Austausch existieren zwar, sind aber in keinem für den Produktiv-Betrieb brauchbaren Zustand
Netzwerk-Management	SNMP wird für alle Funktionen unterstützt	nur wenige Funktionen sind per SNMP steuerbar; Anpassung/Erweiterung des SNMP-Agents möglich
Skalierbarkeit der Netzwerk-Interfaces	max. 6 Karten mit unterschiedlicher Port-Bestückung (z. B. 2xFastEthernet); die Backplane unterstützt max. 1 $\mathrm{GBit/s}$	vom Motherboard abhängige Zahl von Netzwerk-Karten; die Auswahl der unterstützten Schnittstellen-Typen ist beschränkt
Skalierbarkeit der Leistung	abhängig von der *Network Processing Engine* werden unterschiedliche Durchsatzraten unterstützt, max. 1 Mpps; die Speicher-Ausstattung wird dadurch ebenfalls beschränkt und ist bei vielen Routern auf 256 MiByte begrenzt	Beschränkung vor allen Dingen durch die verwendeten Bus-Systeme (PCI) und die Bandbreite des Hauptspeichers; schwankt somit sehr stark in Abhängigkeit vom investierten Geld; Speicher kann in ausreichendem Maße zur Verfügung gestellt werden

Kriterium	Cisco 72xx	Linux mit Quagga
Erweiterbarkeit der Software	hängt stark von den Plänen des Herstellers ab; einige Features sind nur in bestimmten Versions-Strängen verfügbar, die auf Grund anderer Beschränkungen z. T. nicht eingesetzt werden können	leichte Erweiterbarkeit der Funktionalität, wenn entsprechende Software bereits existiert; teilweise ist eigene Software-Entwicklung notwendig; eigene Entwicklungen leicht integrierbar
Stabilität des Betriebs	i. d. R. sehr hoch, wenn ein stabiles IOS-Release verwendet wird; das zuverlässige Prozess-Management sorgt im Falle von Fehlern für Reaktivierung der nötigen Prozesse	stark abhängig von der verwendeten Hardware; für höhere Software-Zuverlässigkeit Überwachung, automatischer Neustart etc. nötig
Aufwand der Administration	durch eine einheitliche Bedien-Oberfläche gering; gute Dokumentation und vielseitiges Lehrgangs-Angebot existiert	durch unterschiedliche Konfigurations-Dateien und -Paradigmen höher; vielseitiges Know-How wird zur Administration benötigt; z. T. schlechte/inaktuelle Dokumentation; komplexe Aufgaben sind leicht automatisierbar
Anschaffungs-Preis mit 2 x FastEthernet	bei Beschaffung auf dem Gebraucht-Markt ab ca. 5000 €; ältere weniger leistungsfähige Modelle z. T. darunter	je nach Ausstattung ab ca. 1000 €

Auch wenn der Linux-basierte Router bei der Hauptaufgabe, dem Paket-Forwarding, etwas schlechter abschneidet, kann er ansonsten gut mit dem Cisco-Router mithalten. Vor allen Dingen lässt sich seine Leistung mit der fortschreitenden Entwicklung der PC-Architektur steigern. Die Möglichkeit in alle Teile der Software-Umgebung einzugreifen sorgt für Flexibilität und Zukunftssicherheit.

Besonders interessant dürfte es sein, die Vorteile beider Plattformen zu vereinen. Der Einsatz einer heterogene Umgebung, z. B. aus einem Cisco-Router und einem Linux-Router, senkt gleichzeitig die Wahrscheinlichkeit eines Total-Ausfalls. Tritt in einem der Systeme ein Fehler auf, wirkt sich dieser nur auf einen Teil der Infrastruktur aus.

Der erhöhte Arbeitsaufwand für einen Router auf PC-Basis lässt sich durch wesentlich geringere Anschaffungskosten und preiswertere Test-Umgebungen sehr gut ausgleichen.

7.2 Zusammenfassung

Im Rahmen dieser Arbeit wurde ein Router auf Basis eines Standard-PCs mit dem Betriebssystem GNU/Linux entwickelt. Das Hauptaugenmerk lag darauf, das neue Internet-Protokoll IPv6 zu unterstützen und eine Lösung für den Einsatz im Umfeld eines Internet-Service-Providers zu schaffen.

Dazu wurden die Anforderungen und vorhandenen Alternativen analysiert und bewertet. Anhand dieser Erkenntnisse wurde ein Anforderungs-Katalog definiert, der die gewünschten Funktionen des Routers enthält.

Anhand dieses Entwurfs wurde der Router implementiert und eine schrittweise Anleitung zur Installation und Konfiguration der notwendigen Software gegeben. Zusätzlich wurde eine Test-Umgebung entwickelt, die umfangreiche Labor-Tests unter geringem Hardware-Aufwand ermöglicht.

Alle im Pflichtenheft aufgestellten Musskriterien konnten erfüllt werden. Die geplanten Wunschkriterien konnten dagegen nicht vollständig umgesetzt werden. Insbesondere die Integration von Multicast-Routing und MPLS konnte mit dem aktuellen Stand der eingesetzten Software nicht vollzogen werden.

Zusammenfassend kann festgestellt werden, dass ein Linux-basierter Router statt eines integrierten Routers eingesetzt werden kann, wenn ein Teil des Komforts verzichtbar und das Know-How zur Unix-Administration vorhanden ist. Auf Grund der günstigen Anschaffungs-Kosten und der guten Interoperabilität sollte ein solcher Router auch zur Erhöhung der Redundanz bereits vorhandener Router in Betracht gezogen werden.

Als größte Vorteile des vorgestellten Systems sind die Flexibilität der Konfiguration und die Skalierbarkeit für sehr komplexe Anwendungsfälle hervorzuheben.

8 Ausblick

Während der hier entwickelte Router auf Basis eines Standard-PCs mit Debian GNU/Linux und Quagga vielen Anforderungen eines Internet-Service-Providers gerecht werden kann, werden in Zukunft wesentlich höhere Bandbreiten benötigt. Um mit dieser Entwicklung Schritt zu halten, wird Hardware-Unterstützung für das Paket-Forwarding unverzichtbar sein. Dieses Ziel verfolgt das Projekt *Liberouter* [Lib] mit seiner *COMBO6* genannten Netzwerk-Interface-Karte. Mit Hilfe dieser freien Entwicklung werden Bandbreiten bis zu 10 GBits mit preiswerten Routern möglich.

Ein anderer Ansatz zur Verbesserung besteht in der Implementation effizienterer Daten-Strukturen für die Routing-Tabelle, die das Paket-Forwarding in einem Software-Router beschleunigt. Dies ist insbesondere für Router in der DFZ wichtig, da hier der von Linux verwendete Routing-Cache mit sehr häufigen Cache-Misses ineffizient arbeitet.

Neben der Verbesserung der Geschwindigkeit müssen aber die wichtigsten Funktionen zur Marktreife geführt werden. Insbesondere für die IPv6-Multicast- und die MPLS-Unterstützung ist noch viel Entwicklungsarbeit nötig, bis diese mit einem Linux-Router verwendet werden können. Auch Quagga bedarf, insbesondere für OSPFv3, noch weitgehender Entwicklung, bis der selbe Funktionsumfang und die selbe Stabilität wie für IPv4 erreicht ist. Gleichzeitig muss eine bessere Integration der Konfiguration aller beteiligten Programme erreicht werden, damit das System gegenüber kommerziellen Routern bezüglich des Administrations-Komfort aufholen kann.

Kurzfristig kann mit Hilfe von DHCPv6 die Automatisierung der Konfiguration von Kunden-Systemen erhöht werden. Mittels *Prefix Delegation* wird die Konfiguration der Kunden-Router erleichtert und somit die Durchdringung des Marktes mit IPv6 auf preiswerte Weise ermöglicht.

Auf Grund von Änderungen in den IPv6-Spezifikationen wird es in Zukunft noch sehr hohen Entwicklungs- und Anpassungs-Bedarf geben. Um auf dem aktuellen Stand der Technik zu bleiben ist es von Vorteil, die Entwicklungen in der *IETF* zu verfolgen.

A Hardware-Voraussetzungen

Die nachfolgenden Hinweise dienen dazu eine möglichst hohe Dienstqualität des Linux-basierten Routers zu gewährleisten. Für kleine Netzwerke reichen aber auch PCs aus, die im Supermarkt angeboten werden.

Allgemein gilt: Je größer die Anforderungen an Verfügbarkeit und Datendurchsatz sind, desto mehr Geld muss auch in einen PC-Router investiert werden.

Gehäuse Da Rechenzentren, in denen ISP ihre Hardware unterbringen, üblicherweise über 19-Zoll-Schränke verfügen und Stellfläche eine teure Ressource ist, muss das Gehäuse diesen Maßen entsprechen. Beim Wählen der Gehäusehöhe konkuriert das Ziel eines geringen Platzverbrauchs mit der Erweiterbarkeit um zusätzlichen Netzwerk-Interface-Karten (NICs).

Da Netzteile erfahrungsgemäß zu den unzuverlässigsten Bauteilen eines Computers gehören, sollten im Betrieb wechselbare redundante Netzteile vorgesehen werden. Teilweise wird mit einer 48 V Stromversorgung gearbeitet. In diesem Fall spielt auch die Verfügbarkeit entsprechender Netzteile eine Rolle. Gehäuse mit einer oder zwei Höheneinheiten kommen hier kaum in Frage.

Motherboard Beim Einsatz von schnellen NICs wird ein mit 33 MHz getakteter 32 Bit breiter PCI-Bus mit einer maximalen Übertragungsrate von 133 $\frac{\text{MByte}}{\text{s}}$, wie er in aktuellen PCs eingesetzt wird, sehr schnell zum Engpass. Auch der aktuell von Gigabit-Ethernet-Karten verwendete 64 Bit PCI-Bus bietet nur wenig Reserven und kann beim Einsatz mehrerer Karten ebenso überlastet werden. Wenn solche Bandbreiten in Zukunft benötigt werden, ist auf ein entsprechend schnelles Bus-System wie *PCI-X* zu achten. Auf dem Motherboard integrierte Netzwerk-Schnittstellen werden teilweise direkt an den Chipsatz angebunden und können die geforderte Bandbreite somit bewältigen. [SR04]

Die von den NICs kommenden Daten müssen durch die Weiterleitungs-Funktion des Betriebssystem-Kerns transferiert werden. Aus diesem Grund muss die Anbindung des Hauptspeichers und der CPU ein Vielfaches des entsprechenden Datendurchsatzes bieten.

CPU Neben dem zu verarbeitenden Datenstrom des Packet-Forwardings muss die CPU noch genügend Reserven zur Berechnung der Routing-Tabellen bieten. Gleichzeitig darf in einem engen Gehäuse nicht zu viel Wärme entstehen. Damit ist der Einsatz einer CPU mit einer Taktfrequenz von mehreren GHz eventuell ungünstig.

Ist neben der reinen Routing-Funktionalität auch der Einsatz von Virtual-Private-Networks (VPNs) mit kryptografischen Verfahren vorgesehen, benötigt dies zusätzliche Rechenzeit. Gerade die in IPSec eingesetzten kryptografisch starken Verfahren lassen sich

gut in Hardware implementieren. Deshalb ist der Einsatz eines entsprechenden Krypto-Beschleuniger-Chips angebracht – als Einsteckkarte, auf dem Motherboard oder direkt in der CPU.

Arbeitsspeicher Da pro Routing-Tabellen-Eintrag ca. 250 Byte [HM01] belegt werden, benötigt ein aktueller Router mindestens 256 MiByte Speicher. Da moderner DDR-RAM[1] sehr erschwinglich ist, spricht nichts gegen den Einsatz größerer Mengen Arbeitsspeicher. Etwa 1 GiByte erscheint zukunftssicher.

Festspeicher Weil ein Router Daten vor allem im Hauptspeicher bearbeitet und nur für das Laden des Betriebssystems und das Speichern der Konfiguration auf Festspeicher zugreifen muss, genügt eine kleine Festplatte. Der Einsatz von Flash-Datenspeicher ist empfehlenswert, denn diese beinhalten keine störanfälligen beweglichen Teile.

Für eine Minimalinstallation reichen ca. 64 MiByte Platz auf dem Datenträger. Eine Größe von 256 MiByte ist auf Grund der gesunkenen Preise für Flash-Speicher zu empfehlen. Damit kann z. B. eine unabhängige Installation als Backup vorgehalten werden, die im Falle von fehlgeschlagenen Änderungen benutzt wird.

Netzwerk-Interface-Karten Bei der Auswahl der Ethernet-NICs ist die von ihnen erzeugte CPU-Last durch Hardware-Interrupts ein wichtiges Kriterium. Desweiteren ist die Treiber-Unterstützung für das Betriebssystem wesentlich. Bewährt haben sich hier insbesondere *Intel EtherExpress Pro 100* bzw. *1000*.

Für andere Netzwerk-Technologien (ATM[2], POS[3] etc.) muss bei der Auswahl der NICs vor allen Dingen auf die Treiber-Verfügbarkeit für das Betriebssystem geachtet werden.

[1] Double Data Rate Speicher – doppelter Datendurchsatz in Bezug auf die Taktfrequenz unter optimalen Bedingungen
[2] Asynchronous Transfer Mode
[3] Packet over SONET/SDH

B Vergleich der IPv4- und IPv6-Header

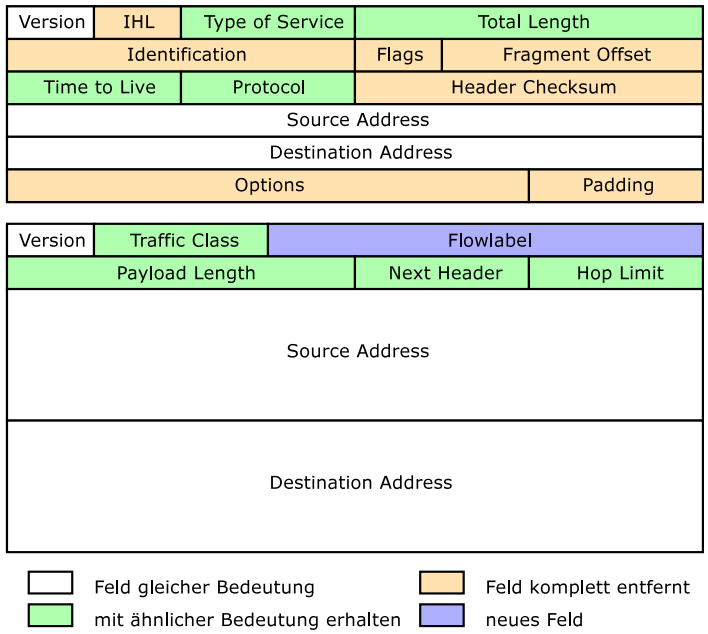

Abbildung B.1: IPv4- und IPv6-Header im Vergleich

Abbildung B.1 zeigt die Änderungen zwischen dem IPv4- und dem IPv6-Paket-Header.

Da IPv6-Header eine feste Länge von 40 Byte und keine Optionen wie bei IPv4 besitzen, benötigen sie auch keine Angabe über die Länge des Headers (IHL) und keine Auffüllung (Padding) auf eine ganzzahlige Menge 32 Bit-Worte.

Der Wegfall der Informationen zur Paket-Fragmentierung (Identification und Fragment Offset) ist auf die zwingende Benutzung von *Path MTU Discovery* (PMTUD) in IPv6 zurückzuführen. Ein Router auf dem Weg zum Ziel braucht also nie zu fragmentieren. Diese Aufgabe wird dem Absender überlassen, der dazu einen entsprechenden Extension-Header benutzt.

Da heutige Übertragungsmedien auf Schicht 2 bereits eine hinreichende Zuverlässigkeit der Übertragung bieten, entfällt die Notwendigkeit den IPv6-Header mit einer Prüfsumme zu versehen. Das Vorhandensein der Prüfsumme führt im Falle von IPv4 bei jedem Zwischensystem, dass die *Time To Live* zwingend um Eins vermindern muss, zu einer entsprechenden Neuberechnung und damit zu Mehraufwand. Auch wenn dieser Vorgang mit Hilfe entsprechender Algorithmen beschleunigt werden kann, stellt eine optionale Absicherung mittels IPSec eine größere Sicherheit dar.

Die Funktion des *Type of Service* wurde bei IPv6 vom Feld *Traffic Class* übernommen. Allerdings ist die Benutzung der *Traffic Class* nicht standardisiert und erfordert eine Koordination zur Festlegung der Bedeutung für die beteiligten Systeme. Deshalb kommt diesem Feld wie bei IPv4 kaum Bedeutung zu.

Statt der *Total Length* über das gesamte IPv4-Paket wird bei IPv6 nur die Länge des nach dem Header transportierten Inhalts angegeben.

Da die *Time to Live* in aktuellen Netzen nichts mehr mit der real vergangenen Laufzeit eines Paketes zu tun hat, wurde diesem Feld der passendere Name *Hop Limit* gegeben. Entsprechend wurde auch das Feld *Protocol* in *Next Header* umbenannt. Ihm kommt allerdings eine zusätzliche Bedeutung für die Erweiterungs-Header zu.

Als einziges wirklich neues Feld wurde das *Flowlabel* eingeführt. Es ist für das Label-Switching in Backbones vorgesehen. Auf Grund der Entwicklung von MPLS kommt dem *Flowlabel* aber derzeit keine praktische Bedeutung zu. In Zukunft kann sich dies ändern – dann ließe sich der zusätzliche Header für das MPLS einsparen.

Trotz all dieser Einsparungen ist der IPv6-Header größer als der von IPv4, da die Absender- und Empfänger-Adressen mit 128 Bit wesentlich länger sind.

C Inhalt der beigefügten CD

Dieser Arbeit ist eine CD-ROM mit zusätzlichen Materialien beigefügt. Die hier aufgeführten Inhalte finden sich auch im Internet unter der Adresse `http://grueneberg.de/andre/diplom/`.

C.1 Verzeichnis-Struktur

`docs/`	Kopien der benutzten frei verfügbaren Literatur
`i-d/`	Internet-Drafts
`rfc/`	RFCs
`source/`	Quelltexte und Patches für die eingesetzte Software
`test_uml.tgz`	Die auf User-Mode-Linux basierende Test-Umgebung (siehe C.2)
`diplom.pdf`	Diese Arbeit im PDF-Format
`config`	Beispiel-Konfiguration eines Linux-Kernels

C.2 Test-Umgebung

Die in Kapitel 6 verwendete Test-Umgebung implementiert mehrere Router auf Basis von virtuellen Maschinen. Dazu wird *User-Mode-Linux*[1] verwendet, mit dem ein Linux-System auf einem anderen simuliert werden kann. Somit lassen sich beliebig komplexe Test-Szenarien auf einer Workstation mit Linux aufbauen. Diese Test-Umgebung ist nicht dafür geeignet, hardwareabhängige oder zeitkritische Tests durchzuführen.

Vor der Nutzung der Test-Umgebung muss die auf dem beigefügten Datenträger enthaltene Datei `test_uml.tgz` mittels

```
tar xzf test_uml.tgz
```

auf der Festplatte entpackt werden. Dabei wird das Verzeichnis `test_uml/` im aktuellen Verzeichnis angelegt. Zusätzlich wird eine lauffähige Installation von *User-Mode-Linux* auf dem Host-System benötigt. Unter Debian steht dazu das Paket `user-mode-linux` zur Verfügung, das mit Hilfe von

```
apt-get install user-mode-linux
```

installiert werden kann.

[1] siehe `http://user-mode-linux.sourceforge.net/`

Im Verzeichnis `test_uml` befinden sich, neben den Root-Dateisystemen für den Management-Server und für fünf Router, die Skripte `startuml` und `mkrouter`. Mit dem Befehl

```
./startuml routerA
```

wird das virtuelle System mit dem Namen `routerA` gestartet. Eine Besonderheit stellt der Management-Server dar, dessen Name `mgmt` lautet, der aber ebenfalls mit diesem Befehl hochgefahren wird.

Jeder virtuelle Host wird von einer eigenen Konsole aus gestartet und ist so konfiguriert, dass er über diese bedient werden kann. Zur Anmeldung am **System** werden die Daten

Benutzer	`root`
Passwort	`diplom`

benötigt. Die Quagga-Daemons können ebenfalls auf die in Kapitel 5 beschriebene Weise bedient werden. Dafür ist das

Quagga-Passwort	`zebra`

notwendig.

Wird ein weiterer Router für andere Test-Szenarien benötigt, kann z. B. mit dem Befehl

```
./mkrouter routerF
```

das Root-Dateisystem für das System `routerF` angelegt werden. Da dieses auf einem generischen Dateisystem aller Router basiert, müssen nach dem ersten Start zuerst alle Netzwerk-Parameter angepasst und der Hostname im virtuellen Linux-System eingestellt werden.

Literaturverzeichnis

[AG02] APITZ, MARIO und GRÜNEBERG, ANDRÉ: *VLAN – Virtual LAN Versuch*. http://www.vlan.ti2000.de/, 2002.

[AGL03] APITZ, MARIO; GRÜNEBERG, ANDRÉ und LÖTZSCH, JANKO: *Netzwerk-Administration – Virtual Private Network mittels IPSec, Beleg*. http://www.ti2000.de/pub/ti2000/6.Semester/NA/vpn-ipsec.pdf, Juli 2003.

[BKR00] BATES, TONY; KATZ, DAVE und REKHTER, YAKOV: *RFC2858 – Multiprotocol Extensions for BGP-4*. http://www.ietf.org/rfc/rfc2858.txt, Juni 2000.

[BM93] BRADNER, SCOTT und MANKIN, ALLISON: *RFC1550 – IP: Next Generation (IPng) White Paper Solicitation*. http://www.ietf.org/rfc/rfc1550.txt, Dezember 1993.

[BM95] BRADNER, SCOTT und MANKIN, ALLISON: *RFC1752 – The Recommendation for the IP Next Generation Protocol*. http://www.ietf.org/rfc/rfc1752.txt, Januar 1995.

[BVL+03] BOUND, JIM; VOLZ, BERNIE; LEMON, TEDD; PERKINS, CHARLES E. und CARNEY, MIKE: *RFC3315 – Dynamic Host Configuration Protocol for IPv6 (DHCPv6)*. http://www.ietf.org/rfc/rfc3315.txt, Juli 2003.

[CER04] *Technical Cyber Security Alert TA04-111A – Vulnerabilities in TCP*. http://www.us-cert.gov/cas/techalerts/TA04-111A.html, April 2004.

[CFM99] COLTUN, ROB; FERGUSON, DENNIS und MOY, JOHN: *RFC2740 – OSPF for IPv6*. http://www.ietf.org/rfc/rfc2740.txt, Dezember 1999.

[Cis] *Cisco Systems, Inc. Homepage*. http://www.cisco.com/.

[Cli] *Click Modular Router Project Homepage*. http://www.pdos.lcs.mit.edu/click/.

[deb03] *The Debian GNU/Linux FAQ*. http://www.debian.org/doc/FAQ/, Februar 2003.

[DH98] DEERING, STEPHEN E. und HINDEN, ROBERT M.: *RFC2460 – Internet Protocol, Version 6 (IPv6) Specification*. http://www.ietf.org/rfc/rfc2460.txt, Dezember 1998.

[Dro03] DROMS, RALPH ET. AL.: *RFC3646 – DNS Configuration options for Dynamic Host Configuration Protocol for IPv6 (DHCPv6)*. http://www.ietf.org/rfc/rfc3646.txt, Dezember 2003.

[Dör04] DÖRING, GERT: *IPv6 BGP filter recommendations*. http://www.space.net/~gert/RIPE/ipv6-filters.html, 2004.

[FMM] FILIP, ONDREJ; MACHEK, PAVEL und MARES, MARTIN: *BIRD User's Guide*. http://bird.network.cz/bird.html.

[HC04] HUITEMA, CHRISTIAN und CARPENTER, BRIAN: *Deprecating Site Local Addresses (draft)*. http://www.ietf.org/internet-drafts/draft-ietf-ipv6-deprecate-site-local-03.txt, März 2004.

[HD03] HINDEN, ROBERT M. und DEERING, STEPHEN E.: *RFC3513 – Internet Protocol Version 6 (IPv6) Addressing Architecture*. http://www.ietf.org/rfc/rfc3513.txt, April 2003.

[HM01] HALABI, BASSAM und MCPHERSON, DANNY: *Internet Routing Architekturen*. Markt und Technik Verlag, München, 2001.

[Hui00] HUITEMA, CHRISTIAN: *Routing in the Internet*. Prentice Hall Ltd., Upper Saddle River, NJ, second edition, 2000.

[IPn] *IPv6 or IPng (IP Next generation)*. http://www.laynetworks.com/IPv6.htm.

[Jun] *Juniper Networks, Inc. Homepage*. http://www.juniper.net/.

[KFH+99] KING, STEVE; FAX, RUTH; HASKIN, DIMITRY; LING, WENKEN; MEEHAN, TOM; FINK, ROBERT und PERKINS, CHARLES E.: *The Case for IPv6*. http://www.ipv6.org/draft-iab-case-for-ipv6-06.txt, Dezember 1999.

[Leu04] LEU, JAMES R.: *MPLS-Linux Mailingliste – Status der Entwicklung von LDP*. http://sourceforge.net/mailarchive/forum.php?thread_id=4811090\&forum_id=5051, Mai 2004.

[Lib] *Liberouter Projekt Homepage*. http://www.liberouter.org/.

[Nex03] *GateD Suite of Routing Protocols*. http://www.nexthop.com/products/datasheets/10.2/overview.pdf, 2003.

[NNS98] NARTEN, THOMAS; NORDMARK, ERIK und SIMPSON, WILLIAM ALLEN: *RFC2461 – Neighbor Discovery for IP Version 6 (IPv6)*. `http://www.ietf.org/rfc/rfc2461.txt`, Dezember 1998.

[Pet03] PETTEY, CHRISTY: *Gartner Says Worldwide Service Provider Router Market Totaled $428 Million in the First Quarter of 2003*. `http://www.gartner.com/5_about/press_releases/pr15may2003b.jsp`, Mai 2003.

[PK94] PARTIDGE, CRAIG und KASTENHOLZ, FRANK: *RFC1726 – Technical Criteria for Choosing IP The Next Generation (IPng)*. `http://www.ietf.org/rfc/rfc1726.txt`, Dezember 1994.

[PPR+99] PALL, GURDEEP SINGH; PALTER, BILL; RUBENS, ALLAN; TOWNSLEY, W. MARK; VALENCIA, ANDREW J. und ZORN, GLEN: *RFC2661 – Layer Two Tunneling Protocol „L2TP“*. `http://www.ietf.org/rfc/rfc2661.txt`, August 1999.

[Qua] *Quagga Homepage*. `http://www.quagga.net/`.

[RL95] REKHTER, YAKOV und LI, TONY: *RFC1771 – A Border Gateway Protocol 4 (BGP-4)*. `http://www.ietf.org/rfc/rfc1771.txt`, März 1995.

[RVC01] ROSEN, ERIC C.; VISWANATHAN, ARUN und CALLON, ROSS: *RFC3031 – Multiprotocol Label Switching Architecture*. `http://www.ietf.org/rfc/rfc3031.txt`, Januar 2001.

[SNM02] *STD62 – Simple Network Management Protocol „SNMP“*. `http://www.ietf.org/rfc/std/std62.txt`, Dezember 2002.

[SR04] SCHMID, PATRICK und ROOS, ACHIM: *Gigabit Ethernet: On-Board Chips Reviewed*. `http://www.tomshardware.com/motherboard/20040430/`, April 2004.

[TD03] TROAN, OLE und DROMS, RALPH: *RFC3633 – IPv6 Prefix Options for Dynamic Host Configuration Protocol (DHCP) version 6*. `http://www.ietf.org/rfc/rfc3633.txt`, Dezember 2003.

[TN98] THOMSON, SUSAN und NARTEN, THOMAS: *RFC2462 – IPv6 Stateless Address Autoconfiguration*. `http://www.ietf.org/rfc/rfc2462.txt`, Dezember 1998.

[USA] *USAGI Project - Linux IPv6 Development Project - Homepage*. `http://www.linux-ipv6.org/`.

[vL01] LEITNER, FELIX VON: *Wegbereitung – MPLS erleichert das Routing im Internet*. c't 8/2001, Seiten 260–262, April 2001.

Eigenständigkeitserklärung

Ich versichere hiermit, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Literatur bzw. Hilfsmittel ohne fremde Hilfe angefertigt habe.

Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt.

Blankenfelde, den 17. Juni 2004

_____

André Grüneberg