Universität Rostock
Fakultät für Ingenieurwissenschaften
Fachbereich Elektrotechnik und Informationstechnik
Institut für Nachrichtentechnik und Informationselektronik

Studienarbeit

Einsatz von IPv6 in modernen Kommunikationsnetzen

cand. ing. Tobias Staudte

13. Februar 2003

Betreuer
Dr.-Ing. H.-D. Melzer
Dipl.-Ing. T. Kessler
Dipl.-Ing. B. Podey

Inhaltsverzeichnis

Inhaltsverzeichnis **5**

1 Motivation **6**

2 Adressdesign **8**

2.1 Adressformat 8

2.2 Unicast, Anycast, Multicast 8

2.2.1 Multicast 9

2.2.2 Unicast- und Anycast-Adressen 9

2.3 IPv4 unter IPv6 und weitere Adressen 11

2.4 DNS 12

3 Der Header **13**

3.1 Aufbau 13

3.2 Erweiterungsheader 15

3.2.1 Routing Header 16

3.2.2 Fragmentation Header 17

3.2.3 Hop-by-Hop Header 18

3.2.4 Destination Header 19

3.2.5 Authentisierung und Verschlüsselung 19

4 ICMP und DHCP **22**

4.1 Aufbau und Typen 22

4.2 Neighbor Discovery Protocol 25

4.2.1 Routererkennung 25

4.2.2 Link-Präfix-Erkennung 25

4.2.3 Address Resolution Protocol 26

4.3 Automatische Adresskonfiguration 28

4.3.1 Stateless Autoconfiguration 28

4.3.2 Stateful Autoconfiguration 29

5 Mobile-IPv6 **31**

5.1 Funktionsweise 31

5.2 Mobile-IPv6 Optionen 32

5.2.1 Home Address 32

5.2.2 Binding Update 33

5.2.3 Binding-Quittung und -Abfrage 34

6 Migration zum neuen Protokoll **36**

6.1 Vorgehensweise 36

6.2 Koexistenz von IPv4 und IPv6 37

6.3 Tunnelverfahren 40

6.3.1 Übersicht 40

6.3.2 IPv6 Tunnel-Broker 41

7 Implementierung **43**

7.1 Linux Client 43

7.1.1 Grundsystem 43

7.1.2 Web-Browser 43

7.1.3 FTP-Client 44

7.2 Linux Server 45

7.2.1 Grundstruktur 45

7.2.2 Webserver Apache 45

7.2.3 FTP-Server PureFTPD 47

7.2.4 Domain Name System 49

7.3 Anbindung an den 6Bone 51

7.3.1 Auswahl des Tunnelproviders und Konfiguration 51

7.3.2 Tunnel Setup Protocol 52

8 Auswertung und Zusammenfassung **55**

A Konfigurationsdatei Tunnelbroker **58**

A.1 freenet6 tspc.conf 58

B Konfigurationsdateien Apache **60**

B.1 Apache Server 60

C Konfigurationsdateien DNS **66**

C.1 /etc/nsswitch.conf 66

C.2 etc/hosts.conf 67

C.3 /etc/hosts 68

C.4 /etc/resolv.conf 69

C.5 /etc/named.conf 70

C.6 Zonendateien 72

C.6.1 Zonendatei 127.0.0.zone 72

C.6.2 Zonendatei localhost.zone 73

C.6.3 Zonendatei 208.30.139.zone 73

C.6.4 Zonendatei ipv4.zone 74

C.6.5 Zonendatei 2001:638:804:8000::204.zone 75

C.6.6 Zonendatei ipv6.zone 76

C.6.7 Zonendatei ::1.zone 77

D Konfigurationsdatei boot.local **78**

E Verzeichnisstrukturen **79**

E.1 Client 79

E.2 Server 79

Abbildungsverzeichnis **83**

Literaturverzeichnis **84**

Kapitel 1

Motivation

Das bestehende Protokoll IPv4 wurde 1973 eingeführt. Damals nicht zum Zweck der Verbindung von kommerziellen und privaten Nutzern, sondern um in den USA die Rechner des Militärs und der Hochschulen miteinander zu verbinden. 1973 wurde die Entscheidung getroffen, die Adressierung mit einer 32-Bit-Adresse durchzuführen. Es zeigte sich bereits früh, dass der dadurch zur Verfügung stehende Adressraum nicht ausreichend würde. Besonders, da man alle Rechner auf der Welt miteinander verbinden wollte. Bereits 1991 wurden Abschätzungen über den in der Zukunft benötigten Adressraumes auf Initiative des IAB (Internet Activities Board) durchgeführt. Die Schätzungen ergaben, dass 2020 etwa 10 Milliarden Menschen auf der Erde leben und jeder statistisch über mehr als einen Computer verfügen wird. Sei es ein Personal Computer, ein Computer im Auto, in Haushaltsgeräten, ein multimediales Mobiltelefon oder ähnliches. Man legte 10^{15} Computer und 10^{12} Netzwerke als Richtwerte für 2020 fest. Tatsächlich belief sich die Zahl der Computer 1996 bereits auf ca. 200 Millionen Stück. Nach Angaben der *IETF* (Internet Engineering Task Force) sollte der Adressraum zwischen 2005 und 2015 den bestehenden Adressraum von ca. 4,3 Milliarden Adressen übersteigen, was dringenden Handlungsbedarf verlangte.

Da das globale Internet sich weiterhin in einem ständigen Wachstum befand und immer noch befindet (Verdoppelung jährlich) und auch die Nutzer-, Dienste-, und Anwendungszahlen immer weiter ansteigen, war ein erweiterter, größerer Adressraum für zukünftige Dienste notwendig. Zusätzlich verknappt die Einteilung des IPv4-Adressraumes in die drei verschiedenen Netz-Klassen (Klasse-A/B/C-Netze) die Anzahl zusätzlich. Bei Anwendung steht der Aspekt der Effizienz im Vordergrund, welche bei IPv4 mit steigender Nutzerzahl stetig abnimmt. Internet-Anwendungen wie Video on Demand, Web-TV oder E-Commerce werden bereits mit IPv4 durchgeführt, jedoch wird die Effizienz durch den Einsatz von IPv6 deutlich verbessert. Ebenfalls wichtige Punkte für die Neuentwicklung eines Protokolls sind die mangelnde Sicherheit beim bestehenden IPv4-Protokoll und die mangelnde Unterstützung von mobilen Geräten. Gerade im Hinblick auf die folgende Mobilfunkgeneration UMTS ist im mobilen Bereich eine schnellere und effizientere Übertragung notwendig. Dies setzt voraus, dass u.a. Routingabläufe verbessert und Routingtabellen verkleinert werden, da diese zum heutigen Zeitpunkt Schwachstellen aufweisen.

Es gab 1992 drei Vorschläge für das neue Internet Protokoll. Zum einen *TUBA* (TCP and UDP over Bigger Address), das auf *CLNP* (Connection-Less Network Protocol) basierte. Dieser Vorschlag wurde verworfen, da es keine Möglichkeiten für Multicast, Mobilität und Ressourcenreservierung gab. Zum Zweiten wurde das *IP Version 7* vorgeschlagen. Es wurde stets weiterentwickelt und wurde 1993 in TP/IX umbenannt. Es ermöglichte eine schnelle Verarbeitung von Datenpaketen und enthielt ein neues Routingprotokoll. Aufgrund der Unvollständigkeit wurde auch dieser Vorschlag verworfen. Als dritter und letzter Vorschlag stand *IP in IP* zur Verfügung. Es beinhaltete zwei Schichten für das Internet-Protokoll. Eine für ein weltweites Backbone und eine für begrenzte Bereiche. 1993 wurde es zu *IPAE* (IP Address Encapsulation) weiterentwickelt. Schließlich entstand *SIP* (Simple IP), das hauptsächlich auf einer Adressvergrößerung beruhte und weiterführende Funktionen, wie Kapselung und optionale Paketfragmentierung besaß. 1993 entstand daraus *SIPP* (Simple IP Plus), da das vorhandene SIP mit dem PIP zusammengeführt wurde. PIP erweiterte SIP mit effektiven Routingstrategien. 1994 überprüfte das IPng-Direktorium alle Vorschläge und entschied sich für ein Protokoll, das auf SIPP basiert. Einzelne Funktionen von SIPP, wie die Adressgröße von 64 auf 128Bit zu vergrößern, wurden umgewandelt. Letztendlich erhielt das neue Protokoll den Namen *IPv6*. Aufgrund der Vergabe der Versionsnummer 5 für ein experimentelles Protokoll, musste das Nachfolgeprotokoll von IPv4 die Version 6 erhalten.

Zusammengefasst treten folgende Probleme bei IPv4 auf:

- 32 Bit Adressbereich zu klein
- Routing ist ineffizient
- mangelnde Unterstützung für mobile Geräte
- mangelnde Unterstützung von real-time Anwendungen
- Schwachstellen hinsichtlich der Sicherheit

Abbildung 1.1: Probleme von IPv4

Es wurden folgende Verbesserungen eingebracht:

- erweiterter Adressraum
- erweiterte Routing-Funktionen
- Vereinfachung der IP-Header-Informationen
- Einführung von Unicast, Anycast und Multicast
- Quality-of-Service
- Unterstützung von Authentifizierung und Sicherheit

Abbildung 1.2: Verbesserungen durch IPv6

Weiterführende Informationen hierzu können in [HU00] und in *IPng-Internet Protocol Next Generation* aus dem Addison-Wesley-Verlag nachgelesen werden.

Kapitel 2

Adressdesign

2.1 Adressformat

Durch die Erweiterung der Adresslänge von 32 Bit auf 128 Bit besteht nun die Möglichkeit, jeden Quadratkilometer der Eroberfläche mit $666 \cdot 10^{21}$ Adressen abzudecken. Durch die Längenänderung entstehen $340 \cdot 10^{35}$ mögliche Adressen.
Die 128 Bit IPv6-Adressen werden durch 16 Bit Integerzahlen dargestellt, wobei jedes Integer aus einem Tupel hexadezimaler Ziffern besteht. Die Tupel werden durch Doppelpunkte voneinander getrennt und führende Nullen müssen nicht angegeben werden. Beispiel: Aus 3ffe:120:210:0:0:0:1 wird somit 3ffe:120:210::1 . Diese Vereinfachung der Schreibweise darf nur einmal bei jeder Adresse verwendet werden, da ansonsten Mehrdeutigkeiten entstehen.
Wie bei IPv4 setzt sich bei IPv6 die Adresse aus einem Präfix und einem lokalen Anteil zusammen wobei die Größe des Präfixes durch einen Schrägstrich mit der Bitanzahl im Anschluss an die Adresse charakterisiert ist. So beinhaltet die Adresse 3ffe:120:210::1/48 einen 48 Bit Präfix (fest) und einen 80 Bit-Anteil für die Vergabe im lokalen Subnetz. Der Präfix wird von links nach rechts festgelegt.

2.2 Unicast, Anycast, Multicast

Bei IPv6 ist die Einteilung in die verschiedenen Netzwerkklassen A, B, C, D und E entfallen. Dieser Schritt ist bei IPv4 durch den Einsatz von CDIR (Classless Internet Domain Routing) und NAT (Network Address Translation) teilweise vollzogen worden. Durch CDIR und NAT wird die zugeteilte Adresse in mehrere Unteradressen aufgesplittet. Ein Router, der einen Rechner im Subnetz erreichen möchte, benötigt nur die Adresse des subnetzverwaltenden Routers. Dieser nimmt die Weiterleitung der Anfrage vor. Diese Art des Subnetting hebt die strikte Klasseneinteilung auf. Die Klasseneinteilung und somit auch das Adressproblem wurde bei IPv4 durch die

Einführung von NAPT (Network Addres Port Translation) stark verbessert. In IPv6 wird der Begriff des Adresstypes eingeführt und man unterscheidet zwischen Unicast-, Anycast- und Multicast-Adressen, wobei der Adresstyp im Adresspräfix definiert ist. Weiterhin ist die von IPv4 bekannte Broadcast-Adresse hier nicht mehr vorgesehen. Für solche Anfragen wird nun die Multicast-Adresse eingesetzt.

2.2.1 Multicast

Eine Multicast-Adresse (s. Abb. 2.1 beschreibt eine Gruppe, die bei einer bestimmten definierten Adresse angesprochen wird. Bei einem "Broadcast" wird ein Datenpaket an alle Schnittstellen übermittelt, die über die angegebene gruppenspezifische Adresse eingetragen sind. Interessant sind Multicast-Adressen für Informationen, die an mehrere Empfänger geichzeitig geschickt werden sollen, wie es unter anderem bei Videokonferenzen, News oder Börseninformationen gewünscht ist. Durch den Präfix 1111 1111 wird auf eine Multicast-Adresse hingedeutet und das

1111 1111	Flag	Gültigkeitsbereich	Gruppen-ID
8 Bit	4 Bit	4 Bit	112 Bit

Abbildung 2.1: Aufbau einer Multicast-Adresse *[RFC2373]*

Flag besteht bis auf das letzte Bit aus Nullen. Durch das letzte Bit wird angezeigt, ob es sich um eine fest vordefinierte (*well-known*) Adresse (Flag=0000) mit bekannter Funktion handelt oder ob es eine frei definierte Adresse (Flag=0001) ist. Die Gruppen-ID kennzeichnet, ob es sich um eine permanente oder um eine vorübergehend gültige Adresse handelt. Damit wird eine weitere Unterteiliung vorgenommen. Multicast-Adressen können weiterhin für bestimmte Gültigkeitsbereiche (*Scope*) eingesetzt werden, was durch die vier Bereichsbits erreicht wird (s. Abb. 2.2).

2.2.2 Unicast- und Anycast-Adressen

Bei IPv6 sind ebenso wie bei IPv4 bestimmte Adressen für spezielle Anwendungen festgelegt. Hierfür dient die Einteilung des Adressraumes. Die Abbildung 2.3 zeigt die Aufteilung des IPv6-Adressraumes in die unterschiedlichen Adresstypen. Die restlichen Adresspräfixe sind noch nicht vergeben. Die Größe des zugewiesenen Adressraumes der restlichen Präfixe ist bereits festgelegt, wodurch später die Adressräume nach Bedarf zugeteilt werden können.

Die *providerbasierte globale Unicast-Adresse* wird für Punkt-zu-Punkt-Verbindungen eingesetzt. Das Problem der großen Anzahl an Routing-Tabellen bei IPv4 sollte durch diesen Adresstyp behoben werden. Man gab dem gesamten Netz eine hierarchische Struktur und ordnete die Adressen nach geographischen Prinzipien zu, wodurch eine lokale Zuordnung der Adressen gewährleistet

Scope	Beschreibung
0000	reserviert
0001	Adresse ist nur innerhalb des lokalen Endgerätes gültig
0010	Adresse ist auf der aktuellen Verbindung oder im lokalen Netzwerk gültig (ähnl. Site-lokal-Unicast)
0101	Adresse soll innerhalb eines local-Bereichs transportiert werden (ähnl Site-lokal-Unicast)
1000	Adresse hat innerhalb einer Firma oder einer Organisation Gültigkeit
1110	Multicasts, die global von Routern in dem Bereich transportiert werden sollen
1111	reserviert
...	weitere Bereiche sind nicht definiert

Abbildung 2.2: Gültigkeitsbereiche einer Multicast-Adresse *[BH01]*

Adresstyp	Präfix	Anteil Adressraum
Reserviert	0000 0000	1/256
Reserviert für NSAP-Zuweisungen	0000 001	1/128
Reserviert für IPX-Zuweisungen	0000 010	1/128
Aggregierbare globale Unicast-Adressen	001	1/8
Providerbasierte globale Unicast-Adresse	010	1/8
Link-Local-Adressen	1111 1110 10	1/1024
Site-Local-Adressen	1111 1110 11	1/1024
Multicast-Adressen	1111 1111	1/256

Abbildung 2.3: Einteilung des Adressraumes *[HU00] [RFC2373]*

werden sollte. Die Hierarchiestufen waren wiefolgt aufgebaut: Internationale Organisationen → Nationale Organisationen → Anbieter → Private Netzstrukturen.

Die *aggregierbare globale Unicast-Adresse* löste die providerbasierte Adresse ab, da die Aufteilung der weltweiten Netze sich nicht an geographischen Grenzen orientiert. Die Netze von unterschiedlichen Betreibern liegen häufig übereinander und parallel zu einander. Daher wurde die providerbasierte Lösung verworfen. Die aggregierbare Adresse erhielt 12,5 Prozent des gesamten Adressraumes, was den größtmöglichen Adressraum darstellte. Sie wird ebenfalls für Punkt-zu-Punkt-Verbindungen verwendet und ermöglicht eine uneingeschränkte Netzanbindung des Rechners. Der gesamte aggregierbare globale Adressraum wird hierarchisch aufgebaut und wird auf *Top-Level-Organisationen*, wie Anbieter der Netz- bzw Internet-Dienste, aufgeteilt. Diese dienen als internationale Verwalter und teilen den zur Verfügung gestellten Adressraum auf die *Next-Level-Organisationen* auf. Dabei handelt es sich um Netz- und Internet-Dienste, die wiederum ihren Adressbereich an individuelle Institutionen weiter aufteilen.

Die *Link-Local Unicast Adressen* besitzt lediglich 0,1 Prozent des Adressraumes. Sie wird innerhalb eines isolierten Subnetzes verwendet, das sie keine Identifikation von Subnetzen enthält. Es handelt sich um eine *unstrukturierte Adresse*, die von angeschlossenen Routern nicht weitergeleitet werden werden, und somit nicht ins Internet gelangen.

Rechnersysteme, auf die von außen kein Zugriff möglich sein soll erhalten eine *Site-Local Unicast Adresse*. Sie ist dem privaten Adressbereich bei IPv4 sehr ähnlich und ermöglicht, innerhalb einer nicht an das globale Netz angeschlossenen Organisation, eine eindeutige Adressierung, ohne dafür global eindeutige Adressen verwenden zu müssen.

Bei *Anycast-Adressen* sind mehrere Endgeräte unter einer Adresse erreichbar. Ein zu übermittelndes Datenpaket wird an eine Adresse mit mehreren Schnittstellen übermittelt, an der anschließend ein Router das Datenpaket an eine einzelne Schnittstelle weiterleitet. Meist ist dies die Schnittstelle, die dem Absender am nächsten liegt. Dies wäre bei einer DNS-Anfrage sinnvoll (DNS-Domain Name System). Anycast-Adressen sind eine spezielle Form der Unicast-Adresse, die zur Adressierung einer einzelnen Schnittstelle an einem Rechner oder Router verwendet wird.

Bei IPv6 kann eine Schnittstelle durchaus mehrere Unicast- sowie auch noch Anycast- und Multicast-Adressen besitzen. Hierdurch ist die Adressierung flexibel einsetzbar und ist effizienter als die IPv4-Adressierung

2.3 IPv4 unter IPv6 und weitere Adressen

Ist keine Adresse vergeben, wird der Router mit einer ”unspezifizierten Adresse“ signalisiert. Sie besteht nur aus Nullen und wird meist bei der Initialisierungsphase verwendet. Die unspezifizierte Adresse sieht demnach wie folgt aus: ’0:0:0:0:0:0:0:0’ beziehungsweise 0::0. Sie dient niemals als Zieladresse, da sie dem Router als unspezifizierte Adresse bekannt ist und somit eine Weiterverarbeitung nicht möglich ist.
Wie bei IPv4 gibt es auch bei IPv6 eine sogenannte ”Loopback-Adresse“. Informationspakete verlassen nie den Rechner und gelangen nicht ins Netz, da sie direkt an die höhere Schicht (OSI) zurückgesendet werden. Die bei IPv4 verwendete 127.0.0.1 wird bei IPv6 durch 0:0:0:0:0:0:0:1 oder ::1 ersetzt. Existieren in einer Netzwerkumgebung Rechner, die IPv6 verarbeiten können und gleichzeitig noch weitere Rechner, die nur IPv4 behandeln können, kann automatisch erkannt werden, welche Schnittstellen nur IPv4 verarbeiten können. Die IPv6-Adresse erhält dann folgenden Aufbau: 0:0:0:0:0:FFFF:a.b.c.d, wobei a.b.c.d die ursprüngliche IPv4-Adresse darstellt. Es handelt sich hierbei um eine auf IPv6 abgebildete IPv4-Adresse. Dies ist durch ein Tunnelverfahren möglich. Die IPv6-adressierten Informationsdaten werden am Netzeingang zum Beispiel durch einen Router komplett mit all seinen Headern in ein IPv4-Paket eingebunden. Die Zieladresse des IPv4-Paketes hat mit der ursprünglichen IPv6-Adresse nichts mehr gemeinsam. Am Ende des Tunnels wird das IPv6-Paket durch Verarbeitung des Headers aus dem IPv4-Paket ausgepackt und weiterverarbeitet. Es ist vorauszusehen, dass in einigen Jahren, wenn IPv6 das hauptsächlich genutzte Protokoll ist, diese Tunnelverfahren umgekehrt eingesetzt werden, um noch vorhandene IPv4-Systeme anzusprechen.

2.4 DNS

Kapitel 3

Der Header

3.1 Aufbau

IPv6 implementiert einen kleinen Header im Vergleich zum IPv4-Header, der durch weitere zusätzliche Header erweitert werden kann. Die Verkleinerung wird dadurch erreicht, dass der von IPv4 bekannte variable Optionsbereich und das 40-Bytes-Limit gestrichen wurde und diese Aufgaben nun durch optionale Header ausgeführt werden. Der ”kleine“ *Basisheader* (s.Abb. 3.2) bleibt immer erhalten und es wird zu den weiteren Headern verwiesen. Jeder Header hat eine festgelegte Funktion und kommt ausschließlich zum Einsatz, wenn er auch benötigt wird. Man nennt dies *verkettete Header*. Weiterhin sind Merkmale wie Header Checksum, Internet Header Length, Fragmentation-, Options- und Padding-Felder bei IPv6 nicht mehr vorhanden. Daher entfällt u.a. die Berechnung der Checksumme, was einen großen Aufwand für die Router darstellte. Die weiteren Aufgaben, wie zum Beispiel die Fragmentierung, werden bei IPv6 nicht mehr vom Router ausgeführt sondern von dem sendenden Rechner bewältigt.

Durch die Vereinfachungen im Header verringert sich die Durchlaufzeit der Pakete durch das Netz wesentlich. Durch die Umstrukturierung des Headers ist eine schnelle Verarbeitung in den einzelnen Stationen möglich und wird Redundanz vermieden, was die Verarbeitungszeit zusätzlich herabsetzt. Da die Router von Aufgaben befreit werden ergibt sich ein weiterer Vorteil. Es entsteht ein schnelleres und effizienteres Protokoll.

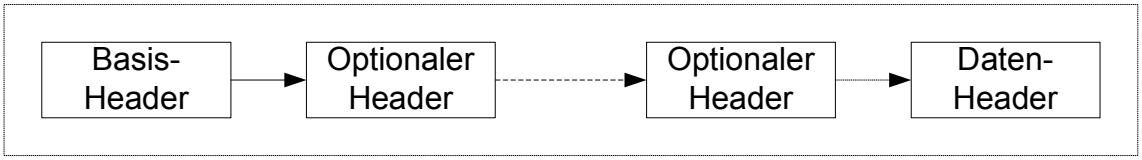


Abbildung 3.1: Prinzip des Headeraufbaus

4 Bit	4 Bit	4 Bit	4 Bit	8 Bit	8 Bit
Version	Class	Flow-label			
Payload Length				Next	Hop-Limit
Source Address					
Source Address					
Source Address					
Source Address					
Destination Address					
Destination Address					
Destination Address					
Destination Address					

Abbildung 3.2: IPv6 Basis-Header *[RFC2460]*

- Version (4 Bit): Unterscheidung der verschiedenen IP-Versionen. Bei IPv6 ist die Zahl 6 eingetragen.
- Class (8 Bit): Bezeichnung der Verkehrsklassen, wodurch eine Priorisierung möglich ist.
- Flow-Label (20 Bit): Dient zur Kennzeichnung der Datenpakete zwischen Sender und Empfänger. Es beinhaltet eine Zufallszahl zwischen 1 und FFFFF(hex) und ist kein Label vorhanden, wird der Wert 0 verwendet. Datenpakete von einem Kommunikationstyp (Sprache, Video, Audio, etc.) besitzen den gleichen Wert. Durch den Flow-Label können die Router dementsprechend reagieren und gesonderte Entscheidungen treffen, so dass zum Beispiel ein gleichmäßiger Datenstrom entsteht. Empfängt ein Router ein gelabeltes Paket legt dieser die gewonnene Ziel- und Absenderadresse zusammen mit dem Label in einer Tabelle ab. Dadurch kann bei einem erneuten Auftreten eines bekannten Labels das Paket weitergeleitet werden, ohne den gesamten Header verarbeiten zu müssen.
- Payload-Length (16 Bit): Gibt die Größe des Paketes, das nach dem IPv6-Header folgt, in Byte an. Durch die 16 Bit ist maximal eine Größe von 65536 Bytes als Angabe möglich. Um größere Pakete zu kennzeichnen wird ein optionaler Header eingefügt und die Payload-Length im Basis-Header wird auf Null gesetzt.

- Next (8 Bit): Dient zur Kennzeichnung des Typs des nächsten Headers nach dem Basis-Header. So kann jede Instanz genau bestimmen, ob es den Header benötigt und sich so durch alle Header durchhangeln. Beinhaltet das Feld den Wert 59, folgt kein weiterer Header.
- Hop-Limit (8 Bit): Gibt die Anzahl von Routern an, die ein Datenpaket durchlaufen darf bis es verworfen wird. Jeder Router dekrementiert den Hop-Limit um einen bis auf Null und sendet nach dem Verwerfen eine Fehlermeldung an den Absender.
- Source-Address (128 Bit): Sendeadresse
- Destination-Address (128 Bit): Empfängeradresse

3.2 Erweiterungsheader

Bei IPv6 werden optionale Erweiterungsheader angefügt, um bestimmte Optionen zu übertragen. So können alle Daten aus dem Basis-Header entfernt werden, die nicht direkt für das Routing notwendig sind und es können weitere optionale Header entwickelt und bei Bedarf eingefügt werden, was die Zukunftssicherheit von IPv6 gewährleistet. Bei den Headern ist eine Reihenfolge (s. Abb. 3.3) festgelegt, damit nicht jeder vorhandene Header durch die Router bearbeitet wird. Der Router kann die Bearbeitung beenden, sobald der in der Reihenfolge nächste für ihn nicht relevante Headertyp vorliegt. Router sind dazu verpflichtet, den Basis-Header, den Destinations-

'Next'-Code	Headertyp
	Basis-Header
0	Hop-by-Hop Options
60	Destination Options Header für den Router
43	Routing Header
44	Fragmentation Header
51	Authentication Header
50	Encapsulating Security Payload Header
60	Destination Options Header für das Endgerät
59	Kein weiterer Header
	Nutzdaten

Abbildung 3.3: Headertypen und Reihenfolge und Typen der Header *[BH01]*

Header für Router und den Routing-Header zu bearbeiten. Alle weiteren Header sind für den Router nicht interessant und müssen nicht untersucht werden.

3.2.1 Routing Header

Der bei IPv6 verwendete Routing-Header übernimmt teilweise die Funktionen der Source Routing Option von IPv4. In diesem Header gibt die Sendestation an, welche Router das Datenpaket bis zum Ziel passieren soll. Der Routing-Header wird von jedem Router abgearbeitet und nach Informationen durchsucht, sofern er vorhanden ist. Der Routing-Header darf maximal 8 Byte lang sein und nur die Felder Next, Length, Typ und Segment sind Pflicht. Alle nachfolgenden Felder sind optional.

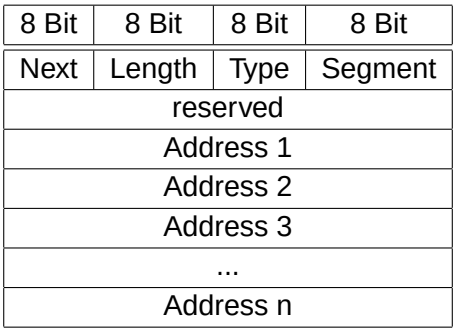


Abbildung 3.4: IPv6 Routing-Header *[RFC2460]*

- Next (8 Bit): Enthält den Typ des nächsten IPv6-Headers
- Length (8 Bit): Enthält die Länge des Erweiterungsheaders, gezählt ab den ersten vier Bytes. Im einfachsten Fall würde hier eine Null stehen.
- Type (8 Bit): Enthält einen Wert zur Unterscheidung des verwendeten Routings. Bisher wird der Wert Null eingetragen.
- Segment (8 Bit): Enthält die Stelle in der Adressenliste, die als nächstes ausgewertet wird. In das Segment-Feld wird beim Beginn der Übertragung die Anzahl der Adressen eingetragen. Enthält dieses Feld den Wert 0, sind alle Routingadressen abgearbeitet und das Ziel ist erreicht.
- Address (32 Bit): Die Adressenliste wird vom Ende zum Anfang hin durch die Router abgearbeitet. Der Router liest das Segment-Feld und das Length-Feld aus, wodurch die nächste Adresse berechnet wird. Diese Adresse wird anschließend mit der Zieladresse im IPv6-Header vertauscht.

3.2.2 Fragmentation Header

Bei großen Datenmengen kann es geschehen, dass ein Paket größer als die *MTU* (Maximum Transfer Unit) ist. Anders als bei IPv4 werden Pakete nicht automatisch fragmentiert. Ist ein Paket zu groß, wird es abgelehnt und eine ICMP-Nachricht (s. Kapitel 4) zurückgesendet, die dem Sender mitteilt, dass das Paket zu groß für die Übertragungsstrecke ist. Der Sender fragmentiert die Daten entsprechend der ICMP-Message und übermittelt mit dem Fragmentation-Header alle benötigten Informationen, damit das Paket vom Empfänger wieder zusammengefügt werden kann. Nach der Fragmentierung wird jedes neu gebildete Paket einzeln geroutet.
Wie erwähnt, nehmen bei IPv6 im Gegensatz zu IPv4 die Sender die Fragmentierung von Daten vor. Bei IPv4 sind für diese Aufgabe die Router zuständig. Eine Ausnahme bei der Fragmentierung stellt der Eingang eines Tunnels dar. In diesem Fall übernimmt der Router die Fragmentierung, um das Paket auf den Tunnel anzupassen. Durch die Auslagerung auf die sendende Station wird eine einfachere und schnellere Implementierung der Router erreicht. Werden bei der Übertragung Netze mit unterschiedlichen MTU´s passiert, sendet der betroffene Router eine ICMP-Nachricht (s. Kapitel 4) an die Sendestation. Diese fragmentiert die Daten erneut, so dass diese den betroffenen Router passieren können. Dieser Ablauf wiederholt sich im ungünstigen Fall mehrmals.

8 Bit	8 Bit	13 Bit	2 Bit	1 Bit
Next	reserviert	Offset	reserviert	M
Identification				

Abbildung 3.5: IPv6 Fragmentation Header *[RFC2460]*

- Next (8 Bit): Enthält den Typ des nächsten IPv6-Headers
- Offset (13 Bit): Enthält in 8-Byte-Einheiten die Stelle, an der dieser Teil der Nutzinformationen in der komletten Nachricht eingesetzt wird.
- M (1 Bit): Ist das M(ore)-Flag auf ’1’ gesetzt folgen weitere Teilblöcke, die zum Vervollständigen der kompletten Nachricht benötigt werden.
- Identification (32 Bit): Enthält einen eindeutigen Wert, damit der Empfänger jedes Teil eindeutig zuordnen kann. Der Wert muss für jedes sich im Transportpfad befindliche Paket unterschiedlich sein. Fragmente, die zu ’einer’ kompletten Nutzinformation gehören besitzen die gleiche Indentifikation.

Basis-Header und alle optionalen Hop-by-Hop-Header, sowie Routing-Header können nicht in Fragmente unterteilt werden. Gehen nicht innerhalb von einer Minute alle zum Vervollständigen der kompletten Nachricht benötigten Pakete beim Empfänger ein, erhält der Sender eine Fehlermeldung.

3.2.3 Hop-by-Hop Header

Durch den Hop-by-Hop-Header werden die gewünschten optionalen Header transportiert, die bei jedem Übertragungsschritt durch den Router ausgewertet werden müssen. Durch den Hop-by-Hop-Header können die Router Pakete selektiv bezüglich einer besonderen Behandlung untersuchen. Dies kann bei speziellen Multicast-Routing-Algorithmen nötig sein, bei denen zum Beispiel Management-Nachrichten authentifiziert sein müssen, so dass sie nur verarbeitet werden können, nachdem der Hop-by-Hop-Header und der Authentifizierungs-Header entfernt wurden. Damit die Router die speziellen Anforderungen erhalten, wird der Hop-by-Hop-Header durch die Router ebenso überprüft wie der Basis-Header (s. Kap. 3.2), um Übertragungsfehler zu vermeiden. Findet eine Auswertung nicht statt, kann es vorkommen, dass der Router von falschen Übertragungsinformationen ausgeht und somit eine fehlerfreie Übertragung des Datenpaketes nicht gewährleistet ist.

Enthält der Basis-Header im Feld 'Next' eine Null, folgt ein Hop-by-Hop-Header. Es sind momentan drei unterschiedliche Zwecke definiert. Zum einen ist es der Jumbo-Payload-Header, Pad1- und der PadN-Header. Ein Header für eine Routeralarmierung ist in Vorbereitung. Erhält ein Router diesen Header, wird das Paket erst komplett verarbeitet, bevor es weitergeleitet wird. Die beiden Pad-Header dienen zum Auffüllen der nicht genutzten Bytes im Header, wobei PadN im Gegensatz zu Pad1 größer als ein Byte sein kann. Wird ein Datenpaket übertragen, das eine Größe von 65536 Bytes überschreitet, kennzeichnet der Jumbo-Payload-Header diesem Umstand durch die Angabe der Größe im Parameterfeld. Der Jumbo-Payload-Header wird nicht verwendet, wenn das Paket einen Fragmentheader enthält.

8 Bit	8 Bit	8 Bit	8 Bit
Next	Length	Type-Options	Length-Options
Parameters			
Parameters			
Parameters			

Abbildung 3.6: IPv6 Hop-by-Hop-Header *[RFC2460]*

- Next (8 Bit): Enthält den Typ des nächsten IPv6-Headers
- Length (8 Bit): Enthält die Größenangabe des nachfolgenden Headers in Byte
- Type-Options (8 Bit): Enthält den Typ der eingefügten Option
- Length-Options (8Bit): Enthält nur die Anzahl der Bytes im Feld 'Parameters'
- Parameters (n Bits): Enthält die gewünschten Optionen und Parameter

Die Größenangabe des nächsten Headers wird durch Vielfache von 8 Bytes angegeben, wobei die ersten 8 Byte nicht mitgezählt werden. Die Mindestgröße des nachfolgenden Headers beträgt 64 Bit. Die Minimalgröße wird folglich durch eine Null im Feld 'Length' signalisiert.
Das Feld 'Lenght-Options' kann ebenfalls eine Null beinhalten. Da hier, im Gegensatz zum Feld 'Length' jedes Byte gezählt wird, ist durch eine Null eine Option ohne Parameter gekennzeichnet. Ist der Wert größer als Null, werden mögliche nicht genutzte Bits und Bytes durch Nullbytes aufgefüllt.

3.2.4 Destination Header

Zusätzlich gewünschte Funktionen werden durch den Destination-Header eingeleitet. Er ermöglicht es, zukünftige Header und Erweiterungen in IPv6 einzubinden. Dadurch ist gewährleistet, dass IPv6 auch bei zukünftigen, bisher unbekannten Diensten, eingesetzt werden kann und das Protokoll den Anforderungen entsprechend angepasst werden kann. Momentan sind für den Destination-Header keine Funktionen definiert. Funktionen wie das Tunneln über IPv6 und die Unterstützung von mobilen Stationen sollen darüber bereitgestellt werden. Der Destination-Header verwendet den gleichen Headeraufbau wie der Hop-by-Hop-Header (s. Abb. 3.6). Der Unterschied besteht darin, dass der Destination-Header im vorangehenden Header mit dem Wert 60 im Feld 'Next' angekündigt wird. Der Destination-Header wird je nach Position in der Headerkette von unterschiedlichen Geräten ausgewertet. Ist der Header vor dem Routing-Header positioniert, wertet jede Station (Transportknoten) auf der Strecke diesen aus. Eine Positionierung nach den Routing-Informationen bewirkt, dass eine Auswertung nur am Ziel erfolgt.

3.2.5 Authentisierung und Verschlüsselung

IPv6 beinhaltet bessere Mechanismen für die Sicherheit im Netz. Dies wird durch Erweiterungsheader erreicht. Da der Transportweg besonderen Gefahren bezüglich Datenverlust, Störungen und Manipulation unterliegt, ist der *Authentication-Header* implementiert worden. Der Absender einer Nachricht ist dadurch eindeutig zu erkennen, und jede Art der Manipulation der Adresse oder des Datenpaketes ist erkennbar. Die Authentisierung garantiert dem Empfänger, dass die Quelladresse authentisch ist und das Paket während der Übertragung nicht verändert wurde. Eine Sicherheit gegen das Mitlesen ist hier nicht vorhanden. Das Mitlesen von Nachrichten, wie es heute unter anderem bei Emails geschieht, kann durch den *Encapsulating Security Payload-Header* verhindert werden. Es wird garantiert, dass nur berechtigte Empfänger den Inhalt des Paketes lesen können.

Der *Authentifizierungsheader* ist ein generischer Header. Meist wird er zwischen den Basis-Header und die Nutzdaten geschrieben. Es sind auch andere Positionen für den Authentifizierungsheader möglich. Er verändert nicht das Verhalten des TCP, des UDP oder des

ICMP. Der Header besteht aus 96 Bits und er enthält die Nummer des nachfolgenden Headers, die Länge der Authentifizierungs-Nutzdaten in 32-Bit-Wörtern, 16 reservierte Bits, ein 32 Bit Sicherheitsparameter-Index (SPI) und ein 32 Bit Folgenummernfeld. Anschließend folgen die Authentifizierungsdaten. Bei IPv4 werden im Gegensatz nur zwei Felder im Header, und zwar die Felder 'Time Tt Live' und 'Header Checksum', für die Authentisierung der Daten verwendet. Es hat sich herausgestellt, dass dieses Verfahren für zukünftige Anwendungen nicht ausreichend ist.

Bei der Authentisierung werden kryptographische Prüfsummen eingesetzt. Zur Anwendung kommen unkomplizierte und schnelle Verfahren. Dabei wird die Prüfsumme über die Nutzdaten, einige Felder des Basis-Headers, den Erweiterungs-Header und einen geheimen Schlüssel gebildet. Je nach eingesetztem Verfahren variiert die Länge der Authentifizierungsnutzdaten. Anhand des Inhalts des Paketes und des SPI berechnet der Empfänger einen Wert für die Authentifzierungsnutzdaten und vergleicht diese mit dem gesendeten Paket. Für die mobile Kommunikation von entfernten Standorten mit dem eigenen Netz ist eine Authentisierung besonders wichtig, damit der Zugang für nicht berechtigte Personen verweigert wird.
Stellt der Empfänger eine Authentisierungsverletzung fest, wird ein Eintrag in der lokalen Log-Datei abgelegt.

Infolge der zusätzlichen Aufgaben der Authentisierung erhöht sich die Wartezeit für die Pakete, da sich die Verarbeitungszeit in den Stationen ebenfalls erhöht. Ein Datenpaket unterliegt somit einer zweifachen Verzögerung, wodurch die Übertragungszeit enorm ansteigt. Die Authentisierung bewirkt jedoch einen deutlichen Sicherheitsgewinn gegenüber IPv4 und ist für die Bedürfnisse der meisten Benutzer ausreichend. Der Sicherheitsgewinn ist so hoch, dass der Nachteil der Verzögerung nicht ins Gewicht fällt.

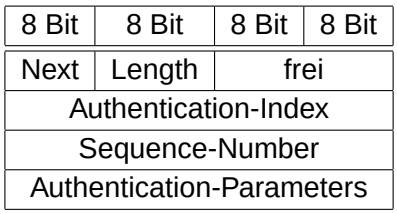


Abbildung 3.7: Authentication-Header *[RFC2402]*

- Next (8 Bit), Lenght (8 Bit): siehe vorherige Erweiterungsheader
- Authentication-Index (32 Bit): Dient zur Verfahrensauswahl. Bei dem Wert Null findet keine Authentisierung statt. Die Werte 1...255 werden je nach IP-Adresse des Zieles vorgegeben. Jeder Router enthält eine Tabelle, in der jeder Adresse ein bestimmtes Verfahren zugeordnet ist.
- Sequence-Number (32 Bit): Enthält fortlaufende Nummer des Nachrichtenpaketes. Es wird bei 1 begonnen und bei jedem Paket um 1 erhöht. Nach der Übertragung von 2^{32} Paketen muss eine neue Sicherheitsverbindung aufgebaut werden.

- Authentication-Parameters (32 Bit): Je nach gewähltem Verfahren werden hier spezifische Parameter hinterlegt.

Da der Authentifizierungsheader keine Daten für unerlaubte Empfänger unleserlich macht ist ein weiteres Verfahren notwendig, um die Daten zu schützen. Es kommt der oben genannte *ESP-Header* (Encapsulating Security Payload-Header) zum Einsatz, der als letzter Header der Erweiterungsheader nach außen sichtbar ist. Zur *Verschlüsselung* werden bei IPv6 zwei Verfahren angewendet. Zum einen werden nur die reinen Nutzdaten verschlüsselt (s. Abb. 3.8) und die restlichen Teile bleiben davon unberührt. So ist es von außen möglich, die Adressen und Optionen zu lesen. Für die meisten Anwendungen ist diese Art der Verschlüsselung ausreichend. Um auch das Auslesen der Adressen und Optionen zu verhindern, kann über einen Tunnel eine Verschlüsselung der gesamten Daten (s. Abb. 3.9) erfolgen. Dabei werden die Daten verschlüsselt und dem neuen Paket wird ein neuer Header vorangestellt. Die zusätzlichen Bits bewirken eine Verringerung der Nettobitrate auf einer Übertragungsstrecke mit festgelegter Bitrate, was wie bei der Authentifizierung aufgrund des hohen Zuwachses an Sicherheit akzeptiert wird.

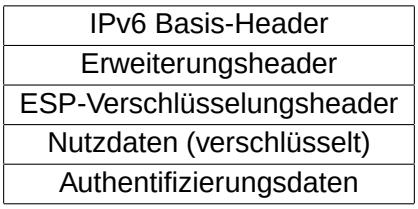


Abbildung 3.8: Beispiel einer Nutzdaten-Verschlüsselung *[DI98]*

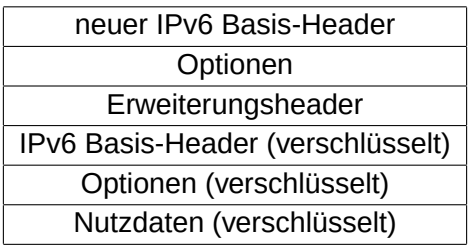


Abbildung 3.9: Tunnel-Verschlüsselung *[DI98]*

Sollte auf dem Transportweg der Header zur Verschlüsselung verloren gehen, ist eine Decapsulation nicht mehr möglich und das gesamte Datenpaket wird verworfen. Um später Untersuchungen zu dem Verlust durchführen zu können, wird jeder Fehler in einer Log-Datei festgehalten.

Weitere Informationen zur Berechnung der Authentifizierungsdaten und zur Verschlüsselung findet man unter anderem bei [HU00].

Kapitel 4

ICMP und DHCP

4.1 Aufbau und Typen

Durch IPv6 ist eine einfache Initialisierung von neu an das Netz angeschlossenen Systemen durchführbar. Das ICMP ('Internet Control Message Protocol') hat ebenso wie bei IPv4 (ICMPv4) die Aufgabe, Fehlermeldungen und Diagnoseinformationen zu übertragen. Die Aufgaben bei IPv6 sind wesentlich umfangreicher, da hier zusätzlich eine automatische Adresskonfiguration unterstützt wird. Hierzu bekommen die Router neue und weitere Funktionen, wodurch sich folglich gewisse Änderungen bei ICMP ergeben. Weiterhin wurden Merkmale, die es bei ICMPv4 gab, die aber nicht mehr genutzt werden entfernt.
Das aus IPv4 bekannte IGMP (Internet Group Management Protocol) wird bei IPv6 als Erweiterung in das ICMPv6-Protokoll eingebunden und fällt weg. Das IGMPv4 besitzt Gruppen-Kontrollfunktionen, mit denen Stationen in eine Zugehörigkeitsgruppe ein- und ausgetragen werden. Da nicht alle IGMPv4-Funktionen bei IPv6 notwendig sind, werden nur die Multicast-Kontrollfunktionen des IGMPv4 in das ICMPv6 eingebunden.
Der aus IPv4 bekannte Dienst ARP (Address Resolution Protocol) zur dynamischen Ermittlung einer MAC-Adresse aus einer gegebenen IP-Adresse wird bei IPv6 vollständig ersetzt. Weiterhin werden neue Pakete zur Übertragung der Adressinformation zwischen Router und Endgerät hinzugefügt.

Siehe hierzu auch RFC2463, in dem das ICMPv6 festgelegt ist.

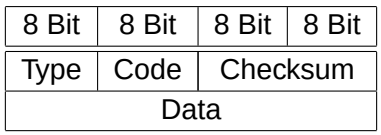


Abbildung 4.1: Aufbau des ICMP-Pakets *[RFC2463]*

- Type: Dient zur Kodierung des ICMP-Pakets und zur Angabe der Funktion der Nachricht.
- Code: Dient zur Variantenunterscheidung der Meldung. Enthält weitere Informationen zu der Nachricht.
- Checksum: Dient zur Feststellung ob die Übertragung korrekt durchgeführt wurde. Zuerst werden die korrekten Werte für die Source- und Target-Adresse bestimmt und anschließend wird die Prüfsumme berechnet.
- Data: Je nach Typ werden hier weitere Informationen zu der Nachricht eingefügt.

Fehlermeldung	**Typ**
Ziel nicht erreicht	1
Paket zu groß	2
Zeitüberwachung abgelaufen	3
Problem mit dem Header	4

Information, Test und Konfiguration	**Typ**
Echo Request/Reply Messages:	
○ Echo-Anforderung	128
○ Echo-Antwort	129
Group Membership Messages:	
○ Anfrage nach Mitgliedschaft in einer Gruppe	130
○ Anzeige einer Gruppenmitgliedschaft	131
○ Beenden der Gruppenmitgliedschaft	132
Autoconfiguration Messages:	
○ Anfrage an Router	133
○ Router-Bekanntmachung	134
○ Anfrage an Nachbar	135
○ Bekanntmachung von Nachbarn	136
○ Umleitung	137

Abbildung 4.2: ICMP-Typen *[RFC2463] [BH01]*

Man unterscheidet bei IPv6-ICMP-Meldungen zwischen den zwei Hauptgruppen (s. Abb. 4.2) „Fehlermeldungen“ und „Informationen, Test und Konfiguration“. Sobald eine IPv6-Station ein Paket verwirft kann es eine ICMP-Nachricht an das ursprüngliche Ziel senden. Dies geschieht allerdings nicht bei Antworten auf Multicast-Paketen und als Antwort auf ICMP-Meldungen, da so Endlosschleifen entstehen können.
Das MSB im Type-Feld besitzt bei Fehlermeldungen eine Null, so dass Fehlermeldungen durch

die Werte 0 bis 127 gekennzeichnet sind. Informationsmeldungen sind durch Werte von 128 bis 255 gekennzeichnet. In der Abbildung 4.2 sind ICMP-Typen dargestellt.

Ein Paket verursacht eine *Typ-1-Nachricht*, wenn eine Firewall das Hineinsenden in ein Netz verhindert. Nachrichten vom *Typ 2* werden gesendet, wenn ein Nachrichtenpaket nicht weitergeleitet werden kann, da es zu groß ist. Über diese Nachricht wird die zulässige Paketgröße auf dem Übertragungsweg ermittelt. Das Ermitteln der MTU wird als *Path MTU-Discovery* bezeichnet. Es wird ein Paket mit der maximalen MTU abgeschickt. Ist das Paket für die Übertragungsstrecke zu groß, sendet die betroffene Station eine Paket-zu-groß-Meldung und zusätzlich die MTU der Verbindung des nächsten Hops. So kann schnell die für die Übertragungsstrecke erforderliche MTU ermittelt werden.
Eine Paket mit abgelaufener Zeitüberwachung, auch *Time Exceed Message* genannt, entsteht sobald das Hop Limit im IPv6-Header an einem Router den Zustand 0 erreicht und dadurch verworfen wird. Der betroffene Router schickt dann eine Nachricht vom Typ 3 an die Station, welche das Paket abgeschickt hat. Wird ein Paket eines fragmentierten Datenstromes verworfen oder bei der Übertragung stark verfälscht, so dass es nicht verwendet werden kann, kann der Datenstrom nicht rekonstruiert werden. Kann die Rekonstruktion nicht innerhalb einer bestimmten Zeit durchgeführt werden, hat dies eine Time Exceed Message zur Folge.
Erhält ein Router oder eine Station ein Datenpaket, so werten diese den Header aus. Hierbei kann ein Problem bei der Auswertung des Basis- oder eines Erweiterungsheaders auftreten, wie es bei fehlerhaften oder unbekannten Parametern im Header vorkommt. Ist dies der Fall wird die Fehlermeldung vom Typ 4 *Parameter Problem Message* an die Quelle des Paketes geschickt, die einen Pointer besitzt, der auf das fehlerhafte Octet-Offset innerhalb des Paketes zeigt. Ursachen für ein Paramter-Problem können ein fehlerhaftes Headerfeld, ein nicht erkannter Nächster-Header-Typ oder eine nicht erkannte IPv6-Option sein.

Zur Überprüfung des Pfades können Diagnosemeldungen verschickt werden. Ein auf dem ICMP-Protokoll basierendes Programm stellt *ping* dar, welches auch bei IPv6 eingesetzt wird. Es schickt eine *Echo Request Message* (Typ 128) an ein vom Programm und vom User definierte Station und erwartet eine *Echo Reply Message* (Typ 129). Auf eine Echo-Anfrage muss jeder IP-Rechner antworten können.
Group Membership Messages enthalten Informationen über die Mitglieder einer Multicast-Gruppe und werden zwischen Multicast-Knoten und Routern ausgetauscht. Nachrichten mit den Typen 133...137 (s. Abb. 4.2) sind unter dem Namen ***Neighbor Discovery*** (s. Kapitel 4.2) zusammengefasst. Sie dienen zur Adresskonfiguration und finden unter anderem bei der *Stateless Autoconfiguration* ihre Anwendung. In der Literatur werden diese auch häufig mit ihrer englischen Bedeutung genannt. Hierbei werden die Begriffe ”Router Solicitation“ (Typ 133), ”Router Advertisement“ (Typ 134), ”Neighbor Solicitation“ (Typ 135), ”Neighbor Advertisement“ (Typ 136) und ”Redirect Message“ (Typ 137) verwendet.

Die *Autoconfiguration Messages* werden im Kapitel 4.2 näher betrachtet.

4.2 Neighbor Discovery Protocol

Die Einführung des *Neighbor Discovery Protokolls* ist aufgrund der gewünschten Plug-and-Play-Fähigkeit von IPv6 zurückzuführen. Es wird als Erweiterung des ICMP-Protokolls verstanden. Explizit ist damit die automatische Konfiguration von Endstationen und die Bestimmung von physikalischen Netzadressen gemeint. Die wichtigsten Funktionen sind das Herausfinden von Routern (*Router Discovery*), die Link-Präfix-Erkennung (*Prefix Discovery*), die Parameter-Ermittlung (*Parameter Discovery*), die automatische Adresskonfiguration und die Ermittlung von physikalischen Netzadressen (*Address Resolution*).

4.2.1 Routererkennung

Das Neighbor-Discovery-Protocol ermöglicht mit der *Router Discovery*, alle an das lokale Subnetz angeschlossenen Router zu entdecken und dessen IPv6-Adresse zu bestimmen. Die Adressen der angeschlossenen Router werden in den Endstationen gespeichert, damit diese Informationen nicht öfters ermittelt werden müssen. Die bekannte Routeradresse ist wichtig, sobald ein Datenpaket aus einem Subnetzwerk heraus gesendet wird. Ist die Zieladresse in einem anderen Subnetzwerk ansässig, so sendet die Quellstation das Datenpaket an einen angeschlossenen Router, was nur durchführbar ist, wenn der sendenden Station die Adresse des Routers bekannt ist. Im Gegensatz zu IPv4 sind bei IPv6 mehrere Router für ein Subnetz zulässig. Der Vorteil der Routererkennung (*Router Discovery*) besteht folglich darin, dass die an das Subnetz angeschlossenen Router durch die Administratoren nicht mehr von Hand eingegeben werden, da eine automatische Erkennung durchgeführt werden kann.
Die an einen Router angeschlossenen Endsysteme fragen mit *Router Solicitation* die Parameter der an das Subnetz angeschlossenen Router ab. Diese antworten mit dem *Router Advertisment* und übermitteln Präfix-Listen für das gesamte angefragte Subnetz an das fragende Endsystem. So haben die Endsysteme die Möglichkeit, ihre Listen abzugleichen und finden auf diesem Wege heraus, welche Rechner als Nachbarn gekennzeichnet werden können.

4.2.2 Link-Präfix-Erkennung

Die *Link-Präfix-Erkennung* dient zur Feststellung, ob sich die Zielstation im gleichen Subnetzwerk befindet. Eine IPv6-Adresse besteht aus einem (128-N)-Bits langen Link-Präfix und einem N-Bits langen Link-Token, wobei der Link-Token zum Beispiel durch eine MAC-Adresse, ISDN-Rufnummer, ATM-Adresse, X.25-Adresse oder ein zufällige Zahl verkörpert wird. Durch den Link-Präfix ist eine Zuordnung des Rechners im Netz durchführbar. Das Subnetzwerk, in dem sich der Rechner befindet, ist mit dem Link-Präfix eindeutig identifiziert. Stationen mit unterschiedlichem Link-Präfix befinden sich nicht im gleichen Subnetz und müssen somit über einen Router angesprochen werden.

Interpretiert ein System eine Zieladresse nicht korrekt und sendet ein für das eigene Subnetz bestimmtes Paket an einen Router, erkennt der Router diesen Fehler. Das betreffende Paket wird mit der *Redirect Function* umadressiert und an den richtigen Rechner weitergeleitet. Zusätzlich erhält der Quellrechner eine Nachricht mit der korrekten Adresse des Zielrechners, damit dieser Fehler nicht erneut auftritt. IPv6 garantiert durch die *Prefix Discovery*-Funktion die Eindeutigkeit des Link-Präfixes in dem Subnetz. Weiterhin muss auch der Link-Token, beziehungsweise die Interface-ID-Adresse eindeutig identifiziert sein. Um dies zu erreichen werden Interface-Adressduplikate mit der *Duplicate Address Detection* erkannt und eine Umadressierung eingeleitet. Da Endsysteme in einem Subnetz nicht dauerhaft am Subnetz angeschlossen sind, ist es notwendig auch die Erreichbarkeit eines Nachbarsystems festzustellen. Bei IPv6 ist diese Funktion in der *Neighbor Unreachability Detection* implementiert.

4.2.3 Address Resolution Protocol

Um bei IPv6 die automatische Konfiguration zu unterstützen, muss das Endsystem Parameter wie den maximalen Hop-Limit (Anzahl der Router, die ein Paket bei einer Übertragung passieren darf) und die MTU-Größe (Maximum Transmission Unit) eines Paketes automatisch ermitteln. Hierzu verwendet die Station die *Parameter Discovery*-Funktion des Neighbor Discovery Protocols von IPv6. Sind alle Parameter ermittelt und der Link-Präfix und Link-Token festgelegt beginnt das Endsystem seinen Sendevorgang. Das Datenpaket wird aus dem Subnetz geschickt und mit Hilfe der Router weitergeleitet. Ist das Paket an dem letzten Router vor dem Ziel angekommen, so muss dieser das Paket an das korrekte Ziel in seinem Subnetz senden. Hierzu benötigt der Router den Link-Token des Zielsystems. Dieser kann bei IPv6 in der Zieladresse angegeben sein. Es können Fehler entstehen, da nicht gewährleistet ist, dass der Link-Token zum Sendezeitpunkt korrekt ist. Dies geschieht, wenn der Zielrechner eine neue Netzwerkkarte erhält, die wiederum eine neue MAC-Adresse besitzt oder wenn das Endsystem in einem verbindungsorientierten Subnetz umgezogen ist. Das Bestimmen der momentanen Netzadresse des Zielsystems wird ähnlich dem ARP (*Address Resolution Protocol*) bei IPv4 durchgeführt. Dort wird eine dynamisch organisierte Adressermittlungstabelle mit IP-Adressen und deren zugehörigen Netzadressen angelegt. Ist die gewünschte Netzadresse nicht vorhanden oder nicht mehr korrekt, so sendet der Router eine Multicast-Nachricht als Broadcast und fragt so die MAC/Netzwerk-Adresse zur gesuchten IP-Adresse an. Der entsprechende Rechner sendet ein ARP-Response mit den Information an den Router zurück, der die neuen Daten in die ARP-Tabelle einträgt und speichert. Bei Ipv6 wird die komplette, korrekte Zieladresse vor dem Senden durch den Quellrechner bestimmt. Hierzu benötigt der Quellrechner allerdings mehr Tabellen als bei IPv4 (s. Abb. 4.3).

Die *Prefix List* beinhaltet die Link-Präfixes aller im lokalen Subnetz angeschlossenen Endsysteme, die über das Router Advertisement bestimmt wurden. Zusätzlich wird jedem Eintrag eine Zeitangabe für die Gültigkeitsdauer des Präfixes zugeordnet. Die zweite Tabelle, der Neighbor Cache, entspricht der ARP-Tabelle bei IPv4 und enthält die IPv6-Adresse und die zugehörigen Link-Adressen der Nachbarstationen. Handelt es sich bei einem Gerät um einen Router, wird ein zusätzliches Flag gesetzt, das R-Flag (*IsRouter*). Aus dem Inhalt des Nachbar-Caches wird

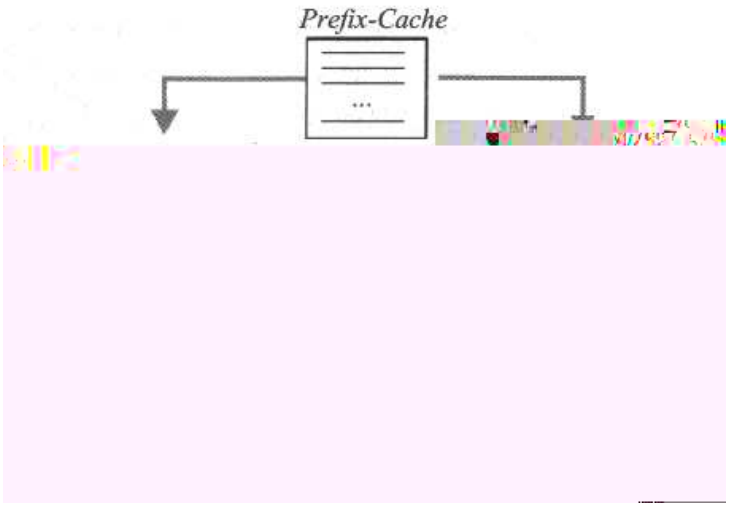


Abbildung 4.3: Caches des Neighbor Protocols *[BH01]*

die dritte Tabelle abgeleitet, der *Router Cache*. Er beinhaltet die Gegenüberstellung Router-IPv6-Adresse und die dazugehörige Link-Adresse für alle am Subnetz angeschlossenen Router. Die letzte Tabelle enthält die Zuordnungen der Ziel-IPv6-Adresse zu Router-IPv6-Adresse. Es können mehrere Router in einem Subnetz angeschlossen sein und es wird jeder Zieladresse ein Router zugeordnet. Soll ein IPv6-Paket übermittelt werden, hält den in Abbildung 4.4 dargestellten Ablauf ein.

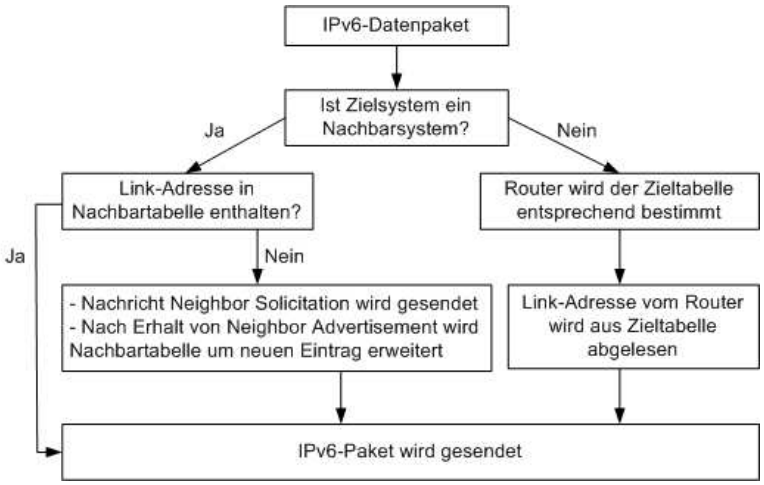


Abbildung 4.4: Ablauf beim Absenden eines IPv6-Paketes im Sender *[BH01]*

4.3 Automatische Adresskonfiguration

4.3.1 Stateless Autoconfiguration

Üblicherweise wird die Link-Präfix-Erkennung zusammen mit der automatischen Adresskonfiguration, bei IPv6 *Stateless Autoconfiguration*, eingesetzt. Wie aus dem Kapitel 4.2 hervorgeht, ist die Stateless Autoconfiguration Bestandteil des *Neighbor Discovery Protocols*. Die Besonderheit bei der Stateless Autoconfiguration besteht darin, dass kein Konfigurationsserver benötigt wird, wenn ein Endsystem in einem Subnetz seine IPv6-Adresse festlegen möchte.

Zur Konfiguration der Adresse sind zwei bedeutende Schritte notwendig. Zuerst muss die eigene Link-Adresse bekannt gemacht und überprüft werden, ob sie nur ein Mal im Subnetz vorkommt. Hierfür sendet die neue Station eine ICMP-Nachricht vom Typ 135 (*Neighbor Solicitation*), die eine Solicited-Mode-Multicast-Adresse enthält. Siehe dazu Abbildung 4.2. Die verwendete spezielle Multicast-Adresse besitzt den 32-Bit Link-Token des Endsystems und einen speziellen 96-Bit Präfix FF02:0:0:0:0:1 und ist nur lokal im eigenen Subnetz gültig und kann als Broadcast-Nachricht verstanden werden. Alle Nachbarsysteme im Subnetz erhalten diese Nachricht. Sobald ein System die angegebene Link-Token-Adresse besitzt, sendet es eine Nachricht vom Typ 136 (*Neighbor Advertisment*) direkt an das fragende System. Wird keine solche Nachricht empfangen, ist das eigene Link-Token als einmalig identifiziert und kann von der Station verwendet werden.
Zweitens erfolgt eine Präfix-Abfrage bei allen angeschlossenen Routern über eine ICMP-Nachricht Typ 133 (*Router Solicitaion*). Die Nachricht enthält wie die Nachricht 135 eine besondere Multicast-Adresse, die als *All-Routers Multicast-Address* bezeichnet wird. Weiterhin enthält die Router Solicitation-Nachricht die im ersten Schritt überprüfte Link-Adresse. Erhält ein Router diese Nachricht als IPv6-Paket, sendet er eine ICMP-Nachricht vom Typ 134 (*Router Advertisement*) mit der aus dem ersten Schritt bekannten Solicited-Mode-Multicast-Adresse und Präfix-Angabe als Option. Somit kennt das gesamte Subnetz die Link-Präfix und das anfragende System. Nun besitzt das neu angeschlossene Endsystem seine Link-Adresse und den Link-Präfix, so dass eine IPv6-Adresse erzeugt werden kann. Das Endsystem kann auch auf die periodisch vom Router gesendete Router Advertisement Nachricht warten.

Es bestehen auch Nachteile bei der Stateless Autoconfiguration. Die zugewiesene IPv6-Adresse ist nur lokal verwendbar, was bedeutet, dass nur in dem betreffenden Subnetz die vergebene Adresse ein Mal vergeben ist, was außerhalb des Subnetzes zu Mehrdeutigkeiten führen kann. Besitzt das Subnetz nur einen Router, so kann bei einem Ausfall keine Stateless Autoconfiguration durchgeführt werden. Es kann nur eine unstrukturierte Link-Local-Adresse gebildet werden, die keine Subnetz-Identifikation enthält. Mit einer unstrukturierten Adresse, Link-Local-Adresse, können keine Informationen aus dem Subnetz gelangen, aber diese Adresse ist für einen internen Datenaustausch verwendbar. Ist nun der vorhandene Router ausgefallen, so kann mit Hilfe der *Stateful Autoconfiguration* ein Link-Präfix angefragt werden. Voraussetzung ist allerdings, dass in dem anfragenden System diese Art der Konfiguration implementiert ist.

4.3.2 Stateful Autoconfiguration

Ebenso wie die Stateless Autoconfiguration ist auch die Stateful Autoconfiguration ein Merkmal des Neighbor Discovery Protocols (s. Kapitel 4.2). Steht kein Router zur Verfügung, übernimmt die Konfiguration der Endsysteme ein zentraler Server. Hierbei handelt es sich um einen DHCP-Server (*Dynamic Host Configuration Protocol*), der ähnlich wie bei IPv4 die Adressen dynamisch verteilt. Da die Adressen aus einem Pool vergeben werden, speichert der Server alle von ihm verteilten Adressen.

DHCP ermöglicht den Hosts, weitere Informationen wie Router-Adressen, DNS-Server-Namen von bestimmten Servern zu erhalten. Damit die Adressenkonfiguration durchgeführt werden kann, sind drei DHCP-Typen definiert: *DHCP-Server*, *DHCP-Client* und *DHCP-Agent*. Der DHCP-Server speichert alle Konfigurationsparameter und ist für die Zuteilung der vorhandenen möglichen Adressen zuständig. Diese werden von einem Endsystem (*DHCP-Client*) angefragt. Wie im Kapitel 4.3.1 bereits beschrieben wird diese Konfigurationsmöglichkeit meist angewendet, wenn kein Router zur Verfügung steht um eine IP-Adresse zu vergeben. Für Netzwerkadministratoren kommt ein weiterer wichtiger Punkt hinzu, eine Stateful Autoconfiguration anstelle einer Stateless Autoconfiguration durchzuführen. Bei der ersten Variante stehen dem Administrator umfangreichere Eingriffsmöglichkeiten zur Verfügung. Da bei der Stateless Autoconfiguration jeder angeschlossene Client vom Router eine IP-Adresse erhält, kann somit auch jeder Client im Subnetz auf das Netz zugreifen. Da dies zu Sicherheitsproblemen führen kann, wird die statuslose Konfiguration verboten, wodurch das Endsystem automatisch die statusbehaftete Autokonfiguration durchführt. Ein Vorteil der statusbehaftete Konfiguration besteht in der subnetzübergreifenden Konfiguration. Da nicht in jedem Subnetz ein DHCP-Server angeschlossen ist, muss mindestens ein Terminal eingerichtet werden, der die Subnetze miteinander verbindet und die Kommunikation zwischen Server und Client ermöglicht. Dieser Terminal wird DHCP-Agent oder DHCP-Relay genannt. Er stellt für die Clients in einem Subnetz den DHCP-Server dar und leitet die Anfragen an einen Server weiter. Der Agent kann sich in einem Router befinden und es können in einem Subnetz mehrere Server und Agents implementiert sein.
Das DHCP-Protokoll hat folgenden Ablauf:
1. Server- bzw. Agentenentdeckung
2. Server- bzw Agentenbekanntmachung
3. Abfrage der Paramter

Der DHCP-Client sucht zu Beginn der Anfrage den im Subnetz angeschlossenen Server oder den angeschlossenen Agenten mit der *DHCP-Solicit*-Nachricht, die alternativ für beide Systeme verwendet werden, da es sich um eine Multicastnachricht handelt. Die Zieladresse lautet FF02::1:3 womit alle DHCP-Server und -Agenten angesprochen werden. Läuft diese Nachricht bei einem DHCP-Server auf, antwortet der Server mit einem *DHCP-Advertise* direkt an den Client und trägt in die Antwort seine IP Server Address und die Client Link-Local Address ein. Der Client beantwortet wiederum diese Nachricht mit einem *DHCP-Request*, um weitere Konfigurationsparamter zu erhalten. Weitere Parameter werden *Extensions* genannt und sie haben die gleiche Bedeutung wie zum Beispiel Informationselemente in ATM-Signalisierungsnachrichten. Der Aufbau eines Ad-

vertise und eines Requests sind prinzipiell sehr ähnlich. Der Request beinhaltet zusätzlich eine Transactions-ID, die eine Kennzeichnung der Client-Server-Beziehung darstellt.

7 Bit	1 Bit	1 Bit	22 Bit
Message Type=2	Flag S	Flag P	Reserved
Server Preference			
Client Link Local IP Address			
Agent IP Address			
Server IP Address			
Extensions			

Abbildung 4.5: DHCP-Advertise *[HU00]*

7 Bit	1 Bit	1 Bit	1 Bit	5 Bit	16 Bit
Message Type=3	Flag C	Flag S	Flag R	Reserved	Transaction-ID
Client Link Local IP Address					
Agent IP Address					
Server IP Address					
Extensions					

Abbildung 4.6: DHCP-Request *[HU00]*

Der DHCP-Server bestätigt das Freigeben der gewünschten Parameter/Extension mit *DHCP-Reply* und sendet diese mit der Nachricht. Eine genauer Beschreibung der Extensions ist dem DRAFT-Dokument zu entnehmen. Befindet sich im eigenen Subnetz kein Server, erreicht die Solicit-Nachricht den Agenten. Dieser antwortet stellvertretend für den Server und teilt dem Client mit Advertise seine Adresse mit. Um dem Client darauf hinzuweisen, dass es sich nicht um den Server handelt ist das S-Flag auf 1 gesetzt. Da die Solicit-Nachricht eine Multicast-Nachricht ist, melden sich alle Agenten bei dem Client an. Da dem Host nun die Agenten bekannt sind sendet er einen Request an einen Agenten, der sich beim Server vorstellt und die benötigten Daten abfragt und den DHCP-Reply vom Server wiederum an den Client weiterleitet.

Ist der Host vollständig konfiguriert können sich die Extensions beim Client oder beim Server ändern. Diese Änderungen werden durch zwei zusätzliche DHCP-Nachrichten signalisiert. Mit der *DHCP-Release*-Nachricht kann ein Client einem Server mitteilen, dass er einige Konfigurationsparameter freigibt, die in den Extensions genauer spezifiziert werden, was durch den Server mit einem DHCP-Reply bestätigt wird. Parameter können sich auch auf der Serverseite ändern, so dass dieser die Möglichkeit besitzt, dem Client die Änderungen mitzuteilen. Dies erfolgt durch die *DHCP-Reconfigure*-Nachricht, die vom Server ausgeht, in der ebenfalls die genauen Änderungen in den Extensions festgehalten sind. Die Release-Nachricht ist vom Aufbau der Advertise-Nachricht sehr ähnlich, wohingegen die Reconfigure-Nachricht sich an der Reply-Nachricht orientiert.

Kapitel 5

Mobile-IPv6

5.1 Funktionsweise

Bei Mobile-IPv6 werden Anwender angesprochen, die häufig ihren Netzstandort ändern. Es können alle Arbeiten und Angelegenheiten erledigt werden, wie es mit einem festen Kommunikationsanschluss möglich ist. Hierzu zählt eine einfache Konfiguration des Zuganges zu den Kommunikations- und Transportwegen. Der Nutzer besitzt an jedem Ort in der Welt die gleiche IP-Adresse. Man beachte, dass bei IPv4 jeder Nutzer eine neue IP-Adresse erhält, sobald er sich aus einem Netz ausloggt und in ein anderes einloggt. Man vergleiche hierzu den Vorgang des Zellenwechsels beim Mobilfunk. Der Teilnehmer behält hier in jeder Kommunikationszelle die gleiche Telefonnummer. Und selbst im Ausland kann er mit der gleichen Rufnummer erreicht werden. Man sieht, dass es ähnliche Ansätze lange in anderen Bereichen gibt, nur dass sie bisher nicht für den Datenaustausch nötig oder nicht durchführbar waren. Dies hat sich geändert. Die fortschreitende Mobilität der Menschen machte es erforderlich, eine Lösung zu finden. IPv6 beseitigt die Engpässe von IPv4 und ermöglicht es, unabhängig vom Standort und ohne komplizierte Konfigurationen, schnell eine Datenkommunikation aufzubauen und durchzuführen.

Eine Station erhält eine feste Adresse und ist diese Station nicht in ihrem Heimatstandort eingeloggt, erhält sie über DHCP oder die statusfreie Adressenauswahl (s. Kapitel 4.3.1) eine zusätzliche lokal gültige Adresse, die *care-of-Adresse*. Diese wird der Station vom angeschlossenen Netz mitgeteilt. Die neue Adresse übermittelt die Station an einen Router im Heimatnetz. Das Heimatnetz leitet über einen Router, der die neue Adresse kennt, die Informationen an die care-of-Adresse. Die Heimatadresse wird in einem Erweiterungsheader eingetragen und dem Paket hinzugefügt. Über Anbindungen mit weiteren Partnern kann das mobile System seine neue Adresse an zusätzliche Router mitteilen. Dadurch wird vermieden, dass die Pakete zuerst zum Heimatnetz geschickt werden.
Anwendungen können die Nachsendeadresse nicht feststellen. IPv6 kopiert die Heimatadresse aus dem Erweiterungsheader in das Paket und gibt erst anschließend das Paket weiter an die

höheren Schichten. Somit ist ein Arbeiten auch aus anderen Netzen für den Anwender möglich. Auch hier kann die Sicherheit durch weitere Header zur Authentisierung gewährleistet werden. Das Heimatnetz erkennt so die Zugehörigkeit der mobilen Station zum Heimatnetz und gewährt den Zugriff auf die Daten und Anwendungen.
Es gibt Überlegungen die Adressbereiche nach Ländern festzulegen oder eine Einteilung der Adressen vorzunehmen. So wäre gewährleistet, dass jeder Teilnehmer seine feste Adresse bekommt, die durch die feste Zuteilung der Adressbereiche einfacher geroutet werden kann. Mobile-IPv6 basiert auf einer gegensätzlichen Idee. Hier erhält der Teilnehmer einer weitere variierende Adresse, was einen Widerspruch zu den bisherigen Überlegungen und Festlegungen darstellt.

5.2 Mobile-IPv6 Optionen

5.2.1 Home Address

Das mobile System kann zusätzlich zu seiner neuen lokalen Adresse auch seine Heimatadresse übermitteln. Hierzu wird ein Destination-Header (s. Kapitel 3.2.4) verwendet, der mit der Option 'Home Address' ausgestattet ist.

8 Bit	8 Bit	8 Bit	8 Bit
Next	Length	Type-Options	Length-Options
Home Adress			

Abbildung 5.1: Destination-Header Home Address *[DI98]*

- Next (8 Bit): Enthält den Typ des nächsten IPv6-Headers
- Length (8 Bit): Enthält die Größenangabe des nachfolgenden Headers in Byte
- Type-Options (8 Bit): Wird mit dem Typ 196 gekennzeichnet
- Length-Options (8Bit): Enthält nur die Anzahl der Bytes im nachfolgenden Feld. (hier Length=16, d.h.: 16*8Bit = 128 Bit)
- Home Address (128 Bits): Enthält die echte Heimatadresse

Jedes von einem mobilen System gesendete Paket muss diesen Header besitzen, da diese Headerinformation nur für das jeweilige Paket gültig ist und nicht gespeichert wird.

5.2.2 Binding Update

Bewegt sich eine Station durch verschiedene Netze, muss sich die Station an dem jeweiligen Netz anmelden. Weiterhin ist es erforderlich, das Heimatnetz von der neuen Adresse in Kenntnis zu setzen, oder die vorherige Nachsendeadresse durch eine neue auszutauschen. Diese Aufgaben werden durch den Destination-Header (s. Kapitel 3.2.4) mit der Option 'Binding Update' ausgeführt.

<table>
<tr><td>1 Bit</td><td>1 Bit</td><td>1 Bit</td><td>5 Bit</td><td>8 Bit</td><td>8 Bit</td><td>8 Bit</td></tr>
<tr><td colspan="4">Next</td><td>Length</td><td>Type-Options</td><td>Length-Options</td></tr>
<tr><td>A</td><td>H</td><td>C</td><td>reserviert</td><td>Prefix</td><td colspan="2">Sequence Number</td></tr>
<tr><td colspan="7">Lifetime</td></tr>
<tr><td colspan="7">Nachsendeadresse</td></tr>
</table>

Abbildung 5.2: Destination-Header Binding Update zum Anmelden *[DI98]*

- Next, Length (je 8 Bit): siehe Kapitel 5.2.1
- Type-Options (8 Bit): Wird mit dem Typ 195 gekennzeichnet
- Length-Options (8Bit): Enthält nur die Anzahl der Bytes im nachfolgenden Feld (hier Length=8 bzw. 24)
- A (1 Bit): Bei gesetztem A-Bit wird eine Quittung für diese Adressenänderung angefordert
- H (1 Bit): Bei gesetztem H-Bit wird der Empfänger ab sofort als Heimatagent eingesetzt
- C (1 Bit): Bei gesetzem C-Bit enthält das letzte Optionsfeld im diesem Header eine gültige Nachsendeadresse
- Präfix (8 Bit): Gibt an, welcher Netzanteil bei der Heimatadresse des rufenden Rechners gültig ist
- Sequence Number (16 Bit): Dient zur Sicherung der Nachricht
- Lifetime (32 Bit): Gibt die Lebensdauer des Eintrages beim Empfänger in Sekunden an
- Nachsendeadresse (128 Bits): Enthält die neue lokale Adresse der mobilen Station

Der empfangende Rechner speichert die Adresse nur, wenn das C-Bit gesetzt ist. Ist das Bit nicht gesetzt, verwendet er die Absenderadresse als Nachsendeadresse. Befindet sich ein Rechner in einem neuen Netz, sendet er einmal mit gesetztem C-Bit damit der Empfänger die neue Adresse speichert. Anschließend mit nicht gesetztem C-Bit, da dem Empfänger die Adresse

schon bekannt ist und die Absenderadresse der Nachsendeadresse entspricht. Um eine vollständige Anmeldung durchführen zu können, muss auch die echte Heimatadresse mit der 'Home Address'-Option übertragen werden.
Enthält das Adressfeld keine Adresse, muss das Längenfeld auf acht gesetzt werden, damit eine eindeutiger Header erzeugt wird. Nur wenn das H-Bit gesetzt ist hat des Feld Präfix eine Bedeutung. Mit dem Feld Präfix kann der Heimatagent aus der Adresse die notwendigen Anycast- und Multicast-Adressen erzeugen. Enthält das Feld Lifetime den Wert 0xffffffff, bleibt die Nachsendeadresse immer gültig. Die Nachsendeadresse wird gelöscht, sobald die Lebensdauer mit 0 angegeben wird oder wenn die Nachsendeadresse und die Heimatadresse identisch ist.

5.2.3 Binding-Quittung und -Abfrage

8 Bit	8 Bit	8 Bit	8 Bit
Next	Length(1)	Typ-Options(1)	Type-Options(2)
Lenght(2)	Status	Sequence Number	
Lifetime			
Refresh			
Adressen weiterer Heimatagenten (optional)			

Tabelle 5.1: Destination-Header Binding-Quittung *[DI98]*

Die Felder für Länge(1) und Type(1) sind identisch mit den Feldern aus dem Kapitel 3.2. Im Feld Type(2) wird der Wert 2 eingetragen und Length(2) enthält analog zu Length(1) die Länge der nachfolgenden Optionen in Bytes. Da die Adressen der Heimatagenten optional sind, wird minimal der Wert 11 (11·8Bit) eingetragen. Die Quittungsmeldung (s. Abb. 5.3) ist im Feld Status untergebracht. Der Empfänger liest die vorherige Sequence-Number aus und trägt diese in den Quittungsheader wieder ein. Weiterhin gibt der Heimatagent bei Lifetime an, wieviele Sekunden er die neue Adresse für gültig erklärt. Die mobile Station muss darauf achten, dass sie diesen Wert gegebenenfalls aktualisiert. Dem mobilen Rechner wird über das Feld Refresh mitgeteilt, nach wievielen Sekunden er die Bindung erneuern soll. Wird die Quittungsanfrage als Anycast durchgeführt, trägt der quittierende Empfänger in dem Adressfeld seine Adresse ein. Um das System auch bei einem Ausfall des Empfängers aufrecht zu halten, ist es dem Empfänger erlaubt, die Adressen zusätzlicher Empfänger einzutragen. Die weiteren eingetragenen Empfänger können dann alternativ zu dem Hauptempfänger als Agenten eingesetzt werden.

Meldung	**Wert**
Bindung wurde akzepiert	0
nicht weiter spezifierter Fehler	128
Fehler im Format der Anforderung	129
Ausführung wurde vom Administrator verboten	130
keine ausreichenden Ressourcen	131
Heimatagenten-Funktion wird nicht unterstützt	132
Rechner liegt nicht im Heimatnetz	133
Sequence Number zu klein	134
Antwort erfolgte auf eine dynamische Anforderung	135
Falsche Länge in der Identifizierung	136
kein Home Agent für diese mobile Station	137

Abbildung 5.3: Quittungsmeldungen Mobile-IPv6 *[IN03]*

Eine *Bindungsabfrage* ist unter anderem interessant für Rechner, die die Nachsendeadresse der mobilen Station nicht kennen. Sie senden eine Option zur Abfrage der momentanen Bindung an die mobile Einheit. Diese sendet an den Absender eine Anmeldung mit seiner aktuellen Bindung. Die Abfrage stellt einen der kleinsten Erweiterungsheader unter IPv6 dar. Die Felder haben eine

8 Bit	8 Bit	8 Bit	8 Bit
Next	Length	Typ-Options	Length-Options

Abbildung 5.4: Destination-Header Bindings-Abfrage *[DI98]*

analoge Bedeutung wie oben betrachtet (s. Abb. 5.1). Die Unterschiede belaufen sich darauf, dass der Typ mit dem Wert 3 deklariert wird. Da es keine weiteren Optionen in diesem Header gibt, muss folglich das Feld Length-Options mit dem Wert 0 ausgefüllt sein.

Kapitel 6

Migration zum neuen Protokoll

6.1 Vorgehensweise

Die Einführung und Umwandlung der bestehenden Systeme auf das neue Protokoll kann aufgrund der hohen Kosten und des großen Aufwands nicht sofort geschehen. Daher wird eine langfristige und kostengünstige Migration durchgeführt, bei der das bestehende Protokoll IPv4 noch lange Zeit neben IPv6 existiert. Dabei werden die Systeme nach Bedarf umgerüstet. Sobald die Möglichkeiten von IPv4 einem System nicht genügen, wird das neue Protokoll eingeführt. Die Endsysteme werden ausgetauscht oder um das neue Protokoll erweitert. Die Umrüstung vollzieht sich über viele Jahre und die Kosten bleiben im Rahmen. Gerade am Anfang der Migration sind nur wenige IPv6-Systeme vorhanden, so dass die Kommunikation hauptsächlich über ein IPv4-Netz durchgeführt wird. Daher ist es sinnvoll, am Anfang bestehende Komponenten nur um IPv6 zu erweitern. Systeme, die IPv4 und IPv6 implementiert haben, werden *Dual-IP-Systeme* genannt.

Die vollständige Umwandlung in ein komplettes weltweites IPv6-Netz geschieht Schritt für Schritt. Zuerst werden DNS-Erweiterungen eingeführt, damit die IPv6-Adressen zugeordnet und bearbeitet werden können und die vorhandenen Endsysteme, Router und Knotenpunkte werden zu Dual-IP-Systeme erweitert. Bei der Erweiterung von DNS ist es nicht ausreichend, die vorhandenen 32 Bit Adressen durch 128 Bit Adressen zu ersetzen, da viele Softwareapplikationen nicht für die Verarbeitung von 128 Bit-Adressen ausgestattet sind. Daher ist es nötig, einen neuen Typ eines Datensatzes zu definieren, eine neue Domain für die Abbildung von Adressen festzulegen und die bereits vorhandenen Routinen zu erweitern. Durch *Translation Router* werden die IP-Header in den benötigten IP-Header umgewandelt. Steht eine ausreichende Anzahl von Translation Routern zur Verfügung ist eine Aufteilung des gesamten Netzes in ein IPv4- und IPv6-Netz möglich. Die anschließende Einführung reiner IPv6-Router erzwingt eine Umstrukturierung des restlichen IPv4-Netzes zum IPv6-Netz, wodurch die Migration abgeschlossen ist.

6.2 Koexistenz von IPv4 und IPv6

Die gleichzeitige Existenz von IPv4 und IPv6 kann aus mehreren Variationen bestehen:
Zum einen können in einem physikalischen LAN IPv4 und IPV6 gleichzeitig existieren (s. Abb. 6.1). Dabei bilden sich zwei logische Subnetze aus, von denen eines das Protokoll IPv4 und

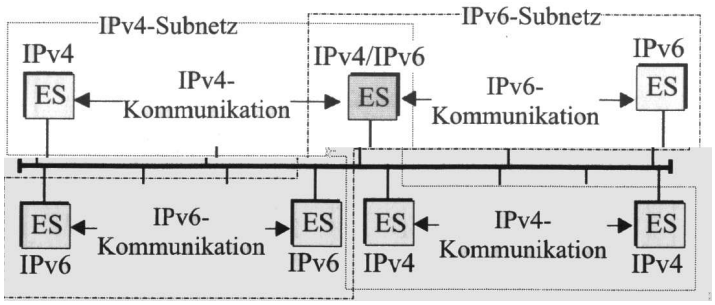
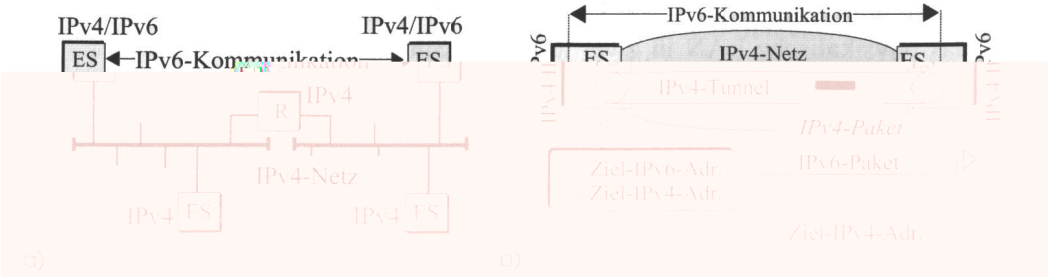


Abbildung 6.1: Paraller Einsatz von IPv4 und IPv6 im LAN *[BH01]*

eines das Protokoll IPv6 verwendet. Bei der Kommunikation zwischen den beiden Dual-IP-Systemen müssen keine IPv4-kompatiblen Adressen verwendet werden, da nur IPv6-Systeme an der Übertragung beteiligt sind. Soll ein IPv4 System angesprochen werden, verwenden die Dual-IP-Systeme das IPv4-Protokoll.

Zum Anderen können sich die Dual-IP-Systeme an einem IPv4-Netz (s. Abb. 6.2) befinden. Dann werden die IPv6-Pakete in einem IPv4-Tunnel untergebracht, um eine Kommunikation zwischen den IPv6-Endsystemen zu ermöglichen. Da das IPv4-Netz keine IPv6-Adressen interpretieren



a)Beispiel einer physikalischen Konfiguration
b)Prinzip der IPv6-Kommunikation

Abbildung 6.2: Dual-IP-Endsysteme am IPv4-Netz *[BH01]*

kann, geschieht in der Quellstation eine Umsetzung der IPv6-Adresse in eine IPv4-Adresse. Da die Gegenseite ebenfalls eine Dual-IP-Station ist, kann als Zieladresse die IPv4-Adresse verwendet werden. Die Zuordnung Ziel-IPv6-Adresse zu Ziel-IPv4-Adresse ist in beiden Dual-IP-Systemen in einer Tabelle abgespeichert.

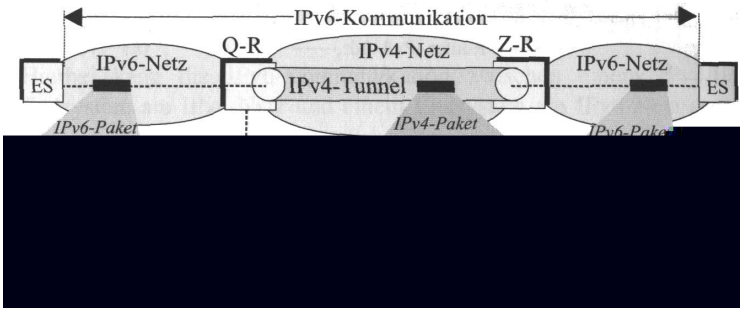
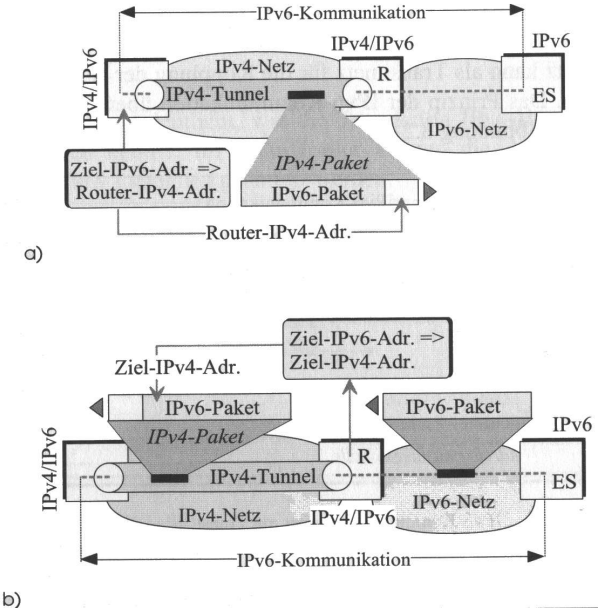


Abbildung 6.3: Kopplung von IPv6-Netzen über ein IPv4-Netz *[BH01]*

Befindet sich zwischen zwei IPv6-Netzen ein IPv4-Netz, das bei einer Kommunikation zwischen den IPv6-Netzen überwunden werden muss (s. Abb. 6.3),setzt sich der Mechanismus mit der Umsetzung der Zieladressen in das benötigte Netzformat und die De- bzw. Encaspulation der IPv4-Tunnel-Pakete an den Routern und Endstationen weiter fort.



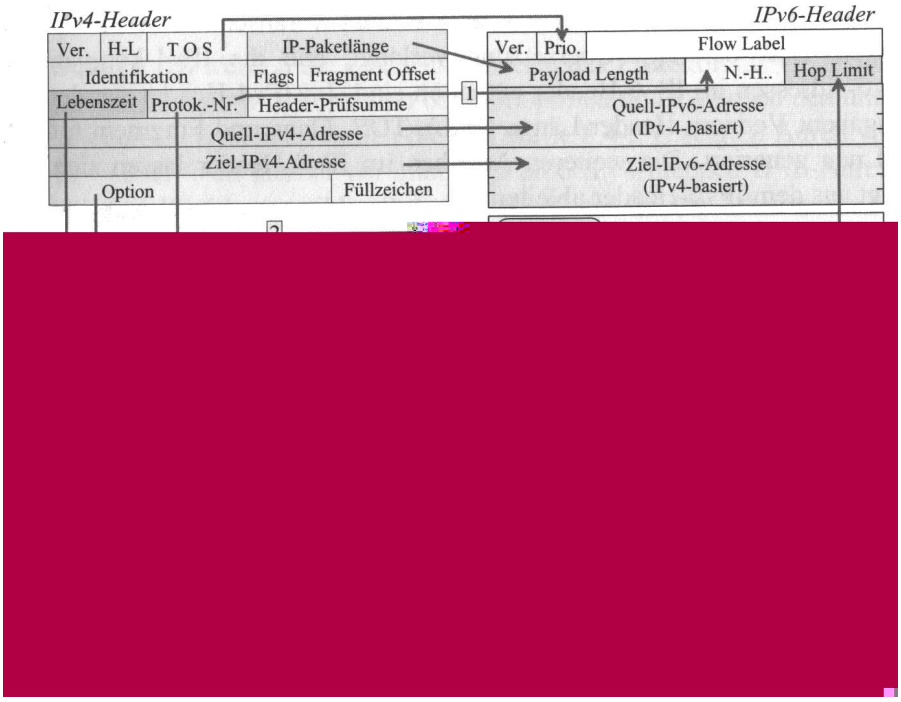
a)Kommunikation über ein IPv4-Netz
b)Kommunikation über ein IPv6-Netz

Abbildung 6.4: Kommunikation bei Erweiterung eines IPv4-Netzes *[BH01]*

Muss die Übertragung zu dem IPv6-Endsystem zuerst über das IPv4-Netz geschehen (s. Abb. 6.4), wird hier die IPv6-Nachricht vom Dual-IP-System in einen IPv4-Tunnel verpackt und an den Router geschickt. Dazu benötigt das Dual-IP-System die Zuordnung der Ziel-IPv6-Adresse zu der entsprechenden Router-IPv4-Adresse, damit dort das Paket weiter bearbeitet wird. Ist das Paket an dem Router angekommen, sendet dieser das Paket über das IPv6-Netz in seiner Ursprungsform mit der gewünschten Ziel-IPv6-Adresse an das Endsystem. Wird die Kommunikation vom

dem reinen IPv6-System initialisiert, sendet es die IPv6-Nachricht an den Router, der ebenfalls eine Tabelle zur Adressumsetzung besitzt. Zusätzlich wird das IPv6-Paket in einen IPv4-Tunnel verpackt und an das Zielsystem geschickt.

Wie oben beschrieben, erfolgt zwangsläufig eine Umsetzung der Adressen und der Nachricht in ein Paketformat, das vom Netz übertragen werden kann. Dazu wird auch der Header umgesetzt, was als *IP-Header-Translation* bezeichnet wird. Die Abbildung 6.5 verdeutlicht die Vorgehensweise. In der Abbildung 6.5a wird ein IPv4-Header in einen IPv6-Header umgesetzt und die Abbildung 6.5b zeigt den Vorgang in umgekehrter Richtung. Die Umsetzung beinhaltet auch die Umwandlung und Ableitung der neuen Daten aus den alten Daten des vorherigen Headers.



a)Umsetzung IPv4-Header in IPv6-Header
b)Umsetzung IPv6-Header in IPv4-Header

Abbildung 6.5: IP-Header-Translation *[BH01]*

6.3 Tunnelverfahren

6.3.1 Übersicht

Das Kapitel 6.2 machte auf die Notwendigkeit von Tunneltechniken aufmerksam, die mit Hilfe der Tunnelendpunkte folgendermaßen eingeteilt werden:
- Router to Router
- Router to Host
- Host to Router
- Host to Host

Wird ein Tunnel zu einem Router benötigt, handelt es sich um *konfigurierbares Tunneling*. Hierbei muss der Router festgelegt sein, zu dem das Paket geschickt wird, sobald mehr als ein Router an das Netz angeschlossen sind. Führt hingegen ein Tunnel zu einem Host und ist die Zieladresse des über das Tunnel zu übertragene IPv6-Paketes IPv4-kompatibel, ist hingegen ein *automatisches Tunneling* notwendig. Das Ende des Tunnels lässt sich aus der IPv4-kompatiblen IPv6-Adresse ableiten. Besonders zu beachten ist bei den Tunneln, dass ihre Verbindung nicht bidirektional verfügbar ist. Sollen Daten in beide Richtungen versendet werden, müssen zwei gegenläufige Tunnel definiert und aufgebaut werden. Die Tunnelstrecke wird als transparente Strecke von den Endpunkten angesehen. Daher wird das Hop Limit nur um einen Wert inkrementiert. Ganz gleich, wie viele Router zwischen den beiden Tunnelendpunkten passiert werden. Bei dem Tunneling sind die ursprünglichen Ziel- und Quelladressen weiterhin im Paket enthalten und werden an den Tunnelanfangs- und -endpunkten ausgewertet.

Am Anfang der Migration wurden die Tunnel noch manuell durch die Administratoren konfiguriert. Da dies einen zu großen Aufwand darstellte, wurde mit unterschiedlichen Verfahren die Konfigurationsarbeit wesentlich verringert. Bei der Verwendung von automatischen Tunneln mit IPv4-kompatiblen Adressen ist frühzeitig eine Möglichkeit gefunden worden, um einzelne Dual-IP Hosts und Routers miteinander zu verbinden. Diese Tunneltechnik verringert nicht das vorhandene Adressproblem. Weiterhin müssten die vollständigen IPv4 Routingtabellen in der IPv6-Umgebung eingebunden werden, was die Anzahl der Tabellen verfünffacht.

Das nachfolgende *6over4* Übertragungsverfahren ist ein Site-Local-Übertragungsverfahren, das auf dem IPv4 Multicast als einen virtuellen Link-Layer basiert.
Bei dem *6to4*-Tunneling wird pro IPv6-Netzwerk nur noch eine IPv4 Adresse für das Tunneling des gesamten Netzwerkes verwendet, was einen speziellen Tunneling Router notwendig macht. Das IPv6-Netz ist über einen speziellen Router an ein IPv4-Netz angeschlossen und besitzt eine IPv4 Adresse, über die der Router erreichbar ist. Zuerst werden die IPv6-Netzwerkpräfixe der angeschlossenen IPv6-Netze gebildet. Sie enthält auch die Ziel-IPv4-Adresse des Routers. Sendet ein Host ein IPv6-Paket an den Router und befindet sich das Ziel nicht im lokalen Netzwerk, entnimmt der lokale Router die Ziel-IPv4-Adresse des Routers, wandelt die Daten in einen IPv4-Tunnel um und verschickt sie der IPv4-Adresse entsprechend zum anderen Netzwerk. Das getunnelte IPv6-Paket enthält dabei auch die Source-Adress des sendenden Netzwerkes, wodurch

beide Netzwerke miteinander kommunizieren können. Der große Vorteil besteht darin, dass nur noch eine IPv4-Adresse benötigt wird, was die Adressenanzahl stark verringert. Ein Nachteil liegt in der vorgegeben starren Struktur des Netzwerkpräfixes und es erfordert Änderungen des IPv6-Präfixes sowie der IPv6 DNS-Einträge, wenn sich die IPv4-Adresse ändert. Es ist ein sinnvolles Verfahren, wenn User mit fester IPv4-Adresse angeschlossen werden sollen.

6.3.2 IPv6 Tunnel-Broker

Ein anderes Verfahren steht mit dem *Tunnel Broker* zur Verfügung. Hier basiert das Verfahren auf einem Server, der ausschließlich für die automatische Abarbeitung von Tunnelanfragen zuständig ist. Man nennt diesen Server *Tunnel-Broker* und das Modell kann als virtueller IPv6 Internet Service Provider betrachtet werden. Den Hosts wird auf diesem Wege ein einfach zu konfigurierender Zugang zu IPv6-Netzen zur Verfügung gestellt, was bei steigender Größe des Netzes immer wichtiger wird. Der Tunnel-Broker überwacht, erstellt, modifiziert und löscht Tunnel, die durch die

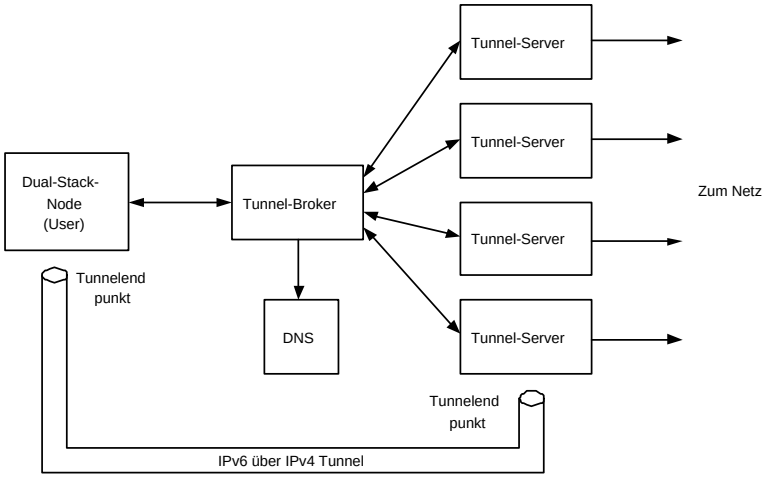


Abbildung 6.6: Tunnel-Broker-Modell *[RFC3053]*

Anwender angefragt werden. Er muss durch die meist vorhandenen IPv4-Nutzer über eine IPv4-Adresse ansprechbar sein. Der Broker ist mit *Tunnel-Servern* verbunden, meist Dual-IP-Router, die den Hosts einen definierten Tunnelendpunkt anbieten und mit dem globalen IPv6-Netz verbunden sind. Es ergibt sich folgender Ablauf. Dual-stack IPv6 Knoten, die an ein IPv4-Netz angeschlossen sind, nehmen mit dem Tunnelbroker Kontakt auf. Dabei tauschen Tunnel-Broker und User Daten zur Identifikation und Authentisierung aus, um eine rechtmäßige Nutzung des Brokers zu überprüfen. Dazu wird auch die IPv4-Adresse des Users, ein Name, DNS-Eintrag des Users und die Userfunktion (Host oder Router) mit dem Tunnel-Broker ausgetauscht. Dieser beauftragt einen angeschlossen Tunnel-Server, als Tunnel-Endpunkt auf der Netzwerkseite

zu fungieren, legt den IPv6-Präfix für den User fest und bestimmt die Lebenszeit für den Tunnel. Weiterhin trägt der Tunnel-Broker die globalen IPv6-Adressen der Tunnel-Endpunkte in den DNS ein. Nachdem diese Konfigurationschritte durchgeführt sind, ist der Tunnel aktiv und dem User werden die nötigen Daten wie Tunnel-Paramter und DNS-Name mitgeteilt. Der User kann nun über einen IPv6-über-IPv4-Tunnel mit dem Tunnelserver kommunizieren, der die Pakete ins Netz verschickt.

Probleme entstehen, wenn die Nutzer einen Dial-Up-Zugang zum Netz verwenden, da hierbei den IPv6-Nutzern bei jeder Einwahl eine neue IPv4-Adresse zugeteilt wird. Dadurch kann ein aufgebauter Tunnel nur für eine kurze Zeit verwendet werden und muss immer wieder neu initialisiert oder aktualisiert werden. Abhilfe schafft ein zusätzliches Protokoll, das zwischen dem Tunnel-Broker und dem User eingesetzt wird. Der User erhält einen *Tunnel-Dämon*, der stellvertretend für den Nutzer die Kommunikation mit dem Tunnel-Broker durchführt und damit eine automatische Konfiguration ermöglicht. Die User-Station erhält bei jedem neuen Verbindungsaufbau die benötigten Daten vom Tunnel-Broker und der Nutzer muss keine manuellen Konfigurationen durchführen.

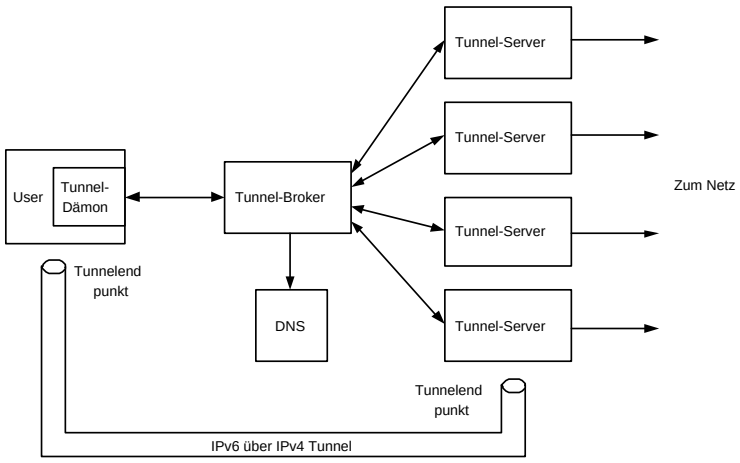


Abbildung 6.7: Tunnel-Broker-Modell mit Dämon *[RFC3053]*

Auf das Tunnel-Broker-Modell mit Tunnel-Dämon wird im Kapitel 7 ausführlicher eingegangen, da es zur Realisierung der Aufgabe verwendet wird.

Kapitel 7

Implementierung

Es wird nun auf eine Beispielimplementierung einer IPv6 Umgebung in ein bestehendes Netz eingegangen. Client und Server werden mit IPv6 Funktionen erweitert und können mit IPv6 Gegenstellen kommunizieren. Eine Übersicht mit IPv6-fähigen Applikationen ist auf den Internetseiten `http://www.ipv6.org/v6-apps.html` und `http://www.bieringer.de/linux/IPv6/status/IPv6+Linux-status-apps.html` zu finden.

7.1 Linux Client

7.1.1 Grundsystem

Als Betriebssystem können IPv6-fähige Linux-Distributionen verwendet werden, wie Debian 2.2, SuSE 7.1, RedHat 7.1 und PLD. Der Client erhält die Distribution Suse Linux 7.3, welche eine Kommunikationsmöglichkeit mit dem 6Bone benötigt.
Der Client ist an einem Subnetzwerk vom Universitätsnetzwerk angeschlossen. Dieses ist so konfiguriert, dass es für den Client wie ein natives IPv6-Netz aussieht, wodurch kein Tunnelverfahren (s. Kapitel 6.3) angewendet werden muss. Um diese Methode der IPv6-Anbindung zu testen, wird der Client im Gegensatz zu dem Server, mit einem Tunnelverfahren ausgestattet. Weitere Einzelheiten sind im Kapitel 7.3.1 beschrieben.

7.1.2 Web-Browser

Um einen Datenaustausch zwischen den beiden Rechner zu realisieren, wird ein Internetbrowser installiert, der IPv6-Funktionen unterstützt. IPv6-fähige Browser sind unter anderem: MS Internet

Explorer 6[1], Mozilla[2], lynx[3] und w3m[4]. Aufgrund der kontinuierlichen Weiterentwicklung und der großen Nutzerzahl wird Mozilla ausgewählt.

Nach der Installation, die in der Dokumentation ausreichend beschrieben wird, kann durch Eingabe einer Adresse eine Internetseite über eine IPv6-Verbindung angezeigt werden. Das Aufrufen kann über einen Domainnamen oder eine IPv6-Adresse erfolgen.

Beispiel Domainname: `http://www.ipv6.comlab.uni-rostock.de`
Beispiel IPv6-Adresse: `http://[2001:638:804:8000::204]`

Bei der IPv6-Adresse ist die geänderte Notation im Vergleich zu IPv4 zu beachten. Es werden bei IPv6 Adressen eckige Klammern verwendet, um dem Browser mitzuteilen, dass es sich um eine IPv6-Adresse handelt und nicht um eine falsch geschriebene IPv4-Adresse. Der Webbrowser kann aufgrund dessen die eingegeben Informationen korrekt auswerten, unterscheiden und verarbeiten.

7.1.3 FTP-Client

Für den Zugang per FTP wird ein IPv6-fähiger FTP-Client benötigt. Die eingesetzte Linux-Distribution besitzt bereits ein IPv6-fähiges FTP-Shell-Skript, über das ein FTP-Zugang realisiert werden kann. Um über IPv6 eine FTP-Verbindung aufzubauen wird in der Shell folgende Eingabe ausgeführt:

```
ftp ftp.ipv6.comlab.uni-rostock.de
```

oder

```
ftp 2001:638:804:8000::204
```

Nach der Eingabe der Benutzerdaten ist eine vollständige IPv6-Connectivity gewährleistet und der Datenaustausch kann durchgeführt werden.

Weitere Applikationen sind nicht notwendig, um einen Client den Datenaustausch mit einem IPv6-Netz und einem IPv6-Rechner zu ermöglichen.

[1] Internetadresse: `http://www.microsoft.de`
[2] Internetadresse: `http://www.mozilla.org`
[3] Internetadresse: `http://lynx.browser.org/`
[4] Internetadresse: `http://ei5nazha.yz.yamagata-u.ac.jp/~aito/w3m/eng/index.html`

7.2 Linux Server

7.2.1 Grundstruktur

Als Betriebssystem kommt beim Server Suse Linux 7.3 zum Einsatz, da es die IPv6-Protokollsuite bereits besitzt und daher leicht zu erweitern ist. Ebenso wie beim Client wird der Webbrowser Mozilla installiert. Da beim Server kein Tunnelverfahren angewendet werden soll, wird der im System befindlichen Netzwerkkarte über den Befehl

```
ifconfig eth0 inet6 add 2001:638:804:8000::204/64
```

die gewünschte IPv6-Adresse zugewiesen und das Default-Gateway mit

```
route --inet6 add default gw  2001:638:804:8000::1
```

bekanntgemacht. Da viele Anwendungen die OpenSSH- und SSL-Library benötigen, werden diese ebenfalls installiert.

7.2.2 Webserver Apache

Als Server wird die kostenlose Applikation Apache 2.0.35 verwendet, welche zusätzlich um die Funktionen PHP4, MySQL und SSL erweitert wird. Es wird Apache ausgewählt, da der vorhandene IPv4-Server auch mit einem Apache-Server ausgestattet ist und dadurch später eine Einbindung des IPv6-Systems leichter durchführbar ist.

Apache 2.0.35

Der Apache-Server 2.0.35 ist bereits mit IPv6-Erweiterungen ausgestattet, so dass keine speziellen IPv6-Konfigurationen durchgeführt werden müssen. Nach dem Download[5] und dem anschließenden Entpacken wird das Installationsskript wie folgt konfiguriert:

```
# ./configure --prefix=/usr/local/apache2 --enable-ssl_module \
              --enable-mod_auth_msql
```

[5] http://www.apache.org

Bei der Konfiguration wird der Installationsroutine mitgeteilt, welcher Installationspfad verwendet werden soll, und dass eine SSL- und MySQL-Erweiterung installiert werden soll. Nach dem Aufruf von '`make`' und '`make install`' ist der Apache-Server im angegeben Verzeichnis installiert und kann gestartet werden. Dazu verwendet man die Eingabe

```
# /usr/local/apache2/bin/apachectl startssl
```

und signalisiert dem Server durch die Angabe von `startssl`, dass eine gesicherte Server-Instanz ausgeführt werden soll.

Um den Apache-Server auf die eigenen Bedürfnisse anzupassen, wird die Datei '`/usr/local/apache2/conf/httpd.conf`' bearbeitet. Hier werden die Angaben für den Serveradministrator und das Dokumentenverzeichnis wie folgt aktualisiert:

```
ServerAdmin staudte@comlab.uni-rostock.de
DocumentRoot "/usr/local/apache2/www"
```

Damit ist die Grundkonfiguration des WWW-Servers abgeschlossen.

PHP4.2.0

Damit der Apache-Server auch PHP4-Internetseiten anzeigen kann, werden die PHP-Erweiterungen in die Serverinstallation eingefügt. Hierzu wird nach dem Download der PHP-Applikation[6] das Paket entpackt und als *Static Module* installiert. Es wird folgende Konfiguration verwendet:

```
# ./configure --prefix=/usr/local/php --with-mysql \
              --with-apache=/usr/local/apache2  \
              --activate-module=modules/php4/libphp4.a
```

PHP wird in das Verzeichnis '`/usr/local/php`' installiert und soll mit dem Apache-Server arbeiten. Die Angabe des Apache-Verzeichnisses bewirkt, dass das Konfigurationsprogramm selbstständig nach der Datei '`httpd.h`' sucht und die erforderlichen Veränderungen vornimmt. Die activte-module-Option bewirkt die Aktivierung der PHP4-Library in der Apache-Konfiguration und der letzte Eintrag nimmt die Erweiterungen für eine Verwendung mit MySQL vor.

Anschließend werden manuell Einträge in die Datei '`httpd.conf`' vorgenommen. Funktionierte die Aktivierung der PHP-Library, befindet sich in der Datei der Eintrag:

[6] `http://www.php.org`

```
LoadModule php4_module modules/libphp4.so
```

Ist dies nicht der Fall, wird er von Hand eingefügt. Zusätzlich werden die Zeilen

```
AddType application/x-httpd-php .php
DirectoryIndex index.php index.php3
```

hinzugefügt oder dementsprechend erweitert.
Nach dem Neustart des Apache-Servers ist der Aufruf und das Ausführen von PHP-Seiten möglich. Damit der Server beim Starten des Rechner automatisch ausgeführt wird, erhält die Datei '`/etc/init.d/boot.local`' (s. Anh. D) den Eintrag

```
/usr/local/apache2/bin/apachectl startssl
```

wodurch der Apache-Server mit SSL-Unterstützung gestartet wird.
Weiterführende Informationen bezüglich des Apache-Servers sind auf der Internetseite `http://www.apache.org` und in der Server-Dokumentation[7] aufgeführt. Die Konfigurationsfiles des Apache-Servers können im Anhang B.1 eingesehen werden.

7.2.3 FTP-Server PureFTPD

Um Nutzern den Datenaustausch mit dem Server zu ermöglichen, wird ein FTP-Server eingerichtet. Aufgrund der gewünschten IPv6-Funktionalität wird der Pure-FTPD-Server installiert. Der Server steht auf der Internetseite '`http://www.pureftpd.org`' zum Download bereit.
Der Server wird mit folgenden Optionen konfiguriert:

```
#./configure --prefix=/usr/local/pure-ftpd --with-puredb \
             --with-language=german --with-everything
```

Die Option - *-with-language=german* installiert den Server in deutscher Sprache, wodurch ein User beim Einloggen einen deutschen Bildschirmtext angezeigt bekommt.
Die Option - *-with-puredb* ist für das gewünschte Authorisierungsverfahren notwendig. Pure-FTPD beinhaltet unterschiedliche Varianten der Authorisierung, wie zum Beispiel über eine MySQL-Datenbank, Puredb oder über das Interface PAM. Bei Puredb werden die Nutzerdaten, wie zum Beispiel Nutzername, Passwort, Home-Verzeichnis, User-Zugehörigkeit, Gruppen-Zugehörigkeit, in das File '`pureftpd.passwd`' geschrieben, welches Pure-FTPD konvertiert, um es verwenden zu können. Das Ändern, Löschen und Hinzufügen von Nutzern kann per Hand

[7] [AB02]

über das File pureftpd.passwd erfolgen, was die Verwaltung der Zugangsberechtigungen einfach gestaltet, oder über die Linux-Konsole.
Die Option - *-with-everything* konfiguriert den Server so, dass ein Großteil der Möglichkeiten installiert werden. Dadurch kann der Server jeweils mit den benötigten Komponenten und Optionen gestartet werden ohne den Server neu zu installieren oder konfigurieren. Für die spätere Verwendung des Servers ist es notwendig, dass *virtuelle User* erlaubt sind. Hierzu im Kapitel 'Userverwaltung' mehr.

Pure-FTPD kann beim Starten mit verschiedenen *Schaltern* gestartet werden. So kann man unter anderem festlegen, ob User auch außerhalb ihrer Home-Verzeichnisse agieren dürfen oder ob die User nur eine bestimmte Speicherplatzkapazität belegen dürfen. Diese Schalter werden in dem *inetd-Serviceprozess* berücksichtigt, über den der FTP-Server gestartet wird. Der inetd-Service setzt Dämons ein, die ihre Dienste über das Netzwerk anbieten. Der inetd-Serviceprozess überwacht die Netzwerk-Ports und startet die Prozesse, sobald sie benötigt werden. Kommt eine Anfrage für einen der überwachten Ports (hier: Port 21=FTP), startet der inetd den zugehörigen Prozess und verbindet den Prozess mit dem Port. Zum Starten des Servers wird der Datei '`/etc/inetd.conf`' der Eintrag[8]

```
ftp stream tcp6 nowait root /usr/local/pure-ftpd/sbin/pure-ftpd
          -B -j -lpuredb:/usr/local/pure-ftpd/etc/pureftpd.pdb
```

hinzugefügt. Besondere Bedeutung besitzt der Wert *tcp6*. Wird für den FTP-Server tcp anstelle von tcp6 eingetragen, ist es dem FTP-Server nicht möglich, IPv6-Anfragen zu bearbeiten, da tcp nur IPv4-Verbindungen bearbeiten kann. Nach dem Starten wird der Server im Hintergrund ausgeführt (Option -B) und es wird bei einem neuen User automatisch ein Verzeichnis angelegt, sobald sich der Nutzer zum ersten Mal auf dem Server einloggt und kein Verzeichnis vorhanden ist (Option -j). Weiterhin wird dem Server die verwendete Authentifizierung (hier: puredb) mitgeteilt (Option -l)und den Pfad mit dem Authentication-File.
Zusätzlich erhält die boot.local-Datei einen Eintrag zum Ausführen des inetd beim Rechnerstart (s. Anh. D).

Eine komplette Übersicht der Schalter und der Konfiguration ist in der Dokumentation und auf der Anbieterseite[9] einsehbar.

Userverwaltung

Um Nutzern, die nicht auf dem lokalen Linux-System als Nutzer vorhanden sind, den Zugang zum FTP-Server zu gewähren, müssen *virtuelle Nutzer* eingerichtet werden. Dazu wird im lokalen System ein neuer Nutzer (userftp) und eine neue Gruppe (groupftp) erzeugt, was mit folgenden Befehlen erfolgt:

[8] Der komplette Eintrag steht in einer Zeile
[9] Internetseite: `http://www.pureftpd.org`

```
groupadd groupftp
useradd -g groupftp -d /dev/null -s /bin/ash ftpuser
```

Virtuelle User können weitere User beinhalten. Dadurch ist der Aufwand bei der Systemkonfiguration gering, da nicht jeder neue User einen Systemzugang benötigt. Nach dem Anlegen des virtuellen Hauptusers und der virtuellen Hauptgruppe wird mit *pure-pw* der Testuser *joe* angelegt. Dazu wird ein virtueller User angelegt, der zu dem Hauptuser *ftpuser* gehört. Weiterhin wird das Home-Verzeichnis festgelegt und mit der Option -m konvertiert Pure-FTPD automatisch die aktualisierte Datei `pureftpd.passwd`:

```
#./pure-pw useradd joe -u ftpuser -m -d /home/ftpusers/joe
```

Nach der Eingabe des Passwortes (hier: *black*) ist dieser Prozess abgeschlossen. Das Passwort wird über einen SSH-Key verschlüsselt und ebenfalls in die zu konvertierende Datei geschrieben. Änderungen an Usern können über

```
#./pure-pw usermod *user*
```

vorgenommen werden. Weitere Informationen zur Userverwaltung können der Dokumentation entnommen werden.

Nachdem der User angelegt wurde, ist ein Zugriff über eine FTP-Software möglich. Der Dokumentation[10] zufolge, können User so konfiguriert werden, dass nur eine Verbindung über SSH erfolgen kann. Leider konnte dieses bisher nicht realisiert werden, was ein Sicherheitsdefizit darstellt.

7.2.4 Domain Name System

Damit die eingerichteten Server über einen Domainnamen erreichbar sind, wird auf dem Serversystem ein *DNS-Server* installiert. Die Grundzüge eines DNS sind im Kapitel 2.4 bereits erläutert worden. Für den DNS-Server wird die Anwendung *Bind-9.2.1*[11] verwendet, welche vollständige IPv6-Unterstützung gewährleistet. Der DNS Bind benötigt eine OpenSSL-Library, die auf der Internetseite: `http://www.openssl.org` kostenlos erhältlich ist und schon installiert wurde. Um eine fehlerlose Verarbeitung von IPv6-Anfragen zu gewährleisten, müssen in der Datei `/etc/nsswitch.conf` die Einträge für `hosts` und `networks` angepasst werden (s. Anh. C.1). Die Installation des DNS ist in der Dokumentation (s. [IS03]) hinreichend beschrieben.

[10] [GU00]

[11] Internetseite: `http://www.isc.org/products/BIND/bind9.html`

Hostkonfiguration

Der konfigurierte DNS-Server ist für die Domains ipv6.comlab.uni-rostock.de und ipv4.comlab.uni-rostock.de verantwortlich. Alle Anfragen bezüglich dieser Domains werden von dem DNS-Server nagilum.comlab.uni-rostock.de an den Server mit der Domain ns1.ipv4.comlab.uni-rostock.de weitergeleitet. Der Server wird daher dementsprechend konfiguriert und erhält den Hostnamen *ns1.ipv4.comlab.uni-rostock.de*. Weiterhin werden in der Datei '`/etc/resolv.conf`' IP-Adressen von weiteren Name-Servern zur Hostnamenauflösung eingetragen.

Die Hostnamenauflösung steuert die Datei '`/etc/host.conf`'. Die Auflösung erfolgt in der angegebenen Reihenfolge, so dass bei der aktuellen Konfiguration (s. Anh. C.2) zuerst die Einträge der lokalen Hosttabelle '`/etc/hosts`' (s. Anh. C.3) untersucht und anschließend die in der Datei '`/etc/resolv.conf`' angegebenen Nameserver befragt werden.

Zonenkonfiguration

Ein DNS ist für selbstdefinierte *Zonen* zuständig, und verarbeitet alle Rechner, die sich in einem bestimmten Domainbereich befinden. Ein DNS-Server ist immer für eine oder mehrere Zonen autorativ. Das Festlegen der Zonen erfolgt in Absprache mit den angeschlossenen Nameservern und deren Administratoren,wobei die Definitionen der Zonen in der Datei '`/etc/named.conf`' stattfindet. Eine Zone benötigt immer zwei Dateien. Eine für die Umsetzung des Domainnamen in eine IP-Adresse und eine weitere für die umgekehrte Umsetzung. Ausgehend von den DNS-Richtlinien entstehen unter anderem für die IPv6-Zone die *Reverse Lookup Zone* '4.0.2.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.8.4.0.8.0.8.3.6.0.1.0.0.2.ip6.int' und die *Domainzone* 'ipv6.comlab.uni-rostock.de'. Durch den entsprechenden Eintrag in '`named.conf`' (s. Ahn.: C.5) werden die Dateien ('`/var/named/ipv6.zone`' und '`/var/named/2001:638:804:8000::204.zone`') aufgerufen, durch die eine weitere Bearbeitung erfolgt. Die Datei '`named.conf`' enthält außerdem Angaben über die Localhost-Zonen, dass der DNS auf IPv6-Anfragen reagieren soll und die weiterleitenden Nameserver.

Durch die Zonedateien in '`/var/named/`' ist festgelegt, dass beim Aufrufen der Domain *www.ipv6.comlab.uni-rostock.de* der Rechner mit dem AAAA-Record *2001:638:804:8000::204* angesprochen wird. Hierbei handelt es sich um den Apache-und FTP-Server. Für weitere Testzwecke wird eine IPv4-Zone eingerichtet, die auf eine IPv4-Adresse verweist, so dass der Server immer erreichbar ist. Die Zonefiles enthalten Angaben über Time To Life, die angebundenen weiteren Nameserver, die Seriennummer der Datei und die Umsetzungsregeln.

Die DNS-Dateien können im Anhang C eingesehen werden.

DNS starten

Nach dem Starten des DNS-Servers mit dem Befehl *named* ist eine Anbindung des DNS-Servers an weitere Server erfolgt. Der Apache-Server und der FTP-Server sind über die Internetadresse `www.ipv6.comlab.uni-rostock` für eine IPv6-Verbindung und über `www.ipv4.comlab.uni-rostock.de` für eine IPv4-Verbindung erreichbar.

Damit der Server, ebenso wie der Apache-Server beim Starten des Rechner automatisch ausgeführt wird, erhält die Datei '`/etc/init.d/boot.local`' (s. Anh. D) einen entsprechenden Eintrag.

Grundlegende Informationen über die Funktionsweise eines Domain Name Systems können in [BH01] und bei [IN03]ausführlich nachgelesen werden.
Informationen zur Installation und Konfiguration des DNS finden sich in der Dokumentation[12] zu *BIND* und bei [MS03].

7.3 Anbindung an den 6Bone

7.3.1 Auswahl des Tunnelproviders und Konfiguration

Die Anbindung an das Testnetz, den *6Bone*, erfolgt beim Linux-Client über einen Tunnelbroker. Siehe auch Kapitel 6.3.2. Zugänge über Anbieter wie Join[13] oder 6bone.net[14] setzen eine feste IP-Adresse voraus, was bei einem mobilen System oder einem Home-System nicht von Vorteil ist, da sich die IP bei jeder Verbindung ändert. Weiterhin ist die Anmeldung bei den letzteren beiden komplizierter und die Serversysteme verfügten im Testzeitraum über keine gute Erreichbarkeit. Der große Vorteil besteht darin, dass bei bei diesem Verfahren über den Anbieter *freenet6*[15] keine feste IP-Adresse benötigt wird. Sollte sich der Client oder der Server später an einem anderen Subnetz befinden, das eine andere IPv4-Adresse besitzt, wird dem Rechner aufgrund der Account Daten wieder die gleiche IPv6-Adresse zugewiesen. Weitere Voraussetzungen für die Verwendung des freenet6-Tunnelbrokers können auf der Internetseite `http://www.freenet6.net` eingesehen werden. Ein Nachteil des Anbieters besteht darin, dass der Tunnel nicht unendlich lange gültig ist. Der Account wird gelöscht, wenn drei Monate lang keine Kommunikation mit dem Tunnelserver stattfindet. Der Service freenet6 bietet eine einfach gestaltete Anmeldung, welche auch für einem nicht versierten Nutzer leicht durchführbar ist. Es werden ein Nutzername und eine email-Adresse verlangt. Nach der erfolgreichen Anmeldung werden die Zugangsdaten per Mail zugeschickt.

[12] [IS03]
[13] Internetadresse: `http://www.join.uni-muenster.de`
[14] Internetadresse: `http://www.6bone.net`
[15] Internetadresse: `http://www.freenet6.net`

Um den IPv6-Tunnel konfigurieren zu können, muss das Linux-System ein spezielles Konfigurationsskript von freenet6 ausführen. Dieses steht auf der Internetseite zur Verfügung und wird heruntergeladen und eingefügt. Nach dem Download muss die Datei "`/bin/hspc.conf`" mit den Zugangsdaten angepasst werden und anschließend erfolgt der Start des Konfigurationsskripts mit "`./hspc -vf hspc.conf`" (s. Anh. A.1). Die von dem Skript durchgeführten Schritte werden auf dem Bildschirm ausgegeben und enthalten unter anderem Informationen über die Tunnelserver IPv4- und IPv6-Adresse, die Host IPv4- Und IPv6-Adresse, die Präfixlänge und den DNS.
Das Skript arbeitet nach dem TSP (Tunnel Setup Protocol), das im Kapitel 7.3.2 näher erläutert wird. Durch die besondere Arbeitsweise von Linux ist der IPv6-Zugang nicht mit Hilfe der angepassten Konfigurationsdatei erfolgt. Linux verwendet zwei Interfaces (SIT0 und SIT1) für den IPv6-Netzzugang über einen Tunnel, die beide konfiguriert werden müssen. Durch eine weitere Änderung in dem Skript, die den Austausch des Interfaces beinhaltet und eine anschließende Abspeicherung unter dem Namen "`hspc2.conf`" wird dieses Problem behoben. Bei der Initialisierung wird zuerst die Konfigurationsdatei mit dem Interface SIT0 und anschließend die mit dem Interface SIT1 ausgeführt. Der Rechner wird automatisch konfiguriert, erhält eine IPv6-Adresse, ist mit einem Tunnel ausgestattet und hat volle IPv6 Connectivity, so dass eine Kommunikation mit dem 6Bone möglich ist.

7.3.2 Tunnel Setup Protocol

Der Service Dienst freenet6 verwendet das *TSP* (Tunnel Setup Protocol), das von der Betreiberfirma Viagenie entwickelt wurde, zur Vergabe von Adressen. Es automatisiert den Einsatz von IPv6 über IPv4, unter Zuhilfenahme von konfigurierten Tunneln. TSP baut auf dem vorhandenen TCP (Transport Control Protocol) auf und verwendet für Kommunikation XML-Nachrichten. TCP ist ein verbindungsorientiertes Protokoll, wobei der aufgebaute Kanal über Mechanismen, wie Zeitüberwachung, Quittierung und Prüfsumme verfügt. Diese dienen zur Überprüfung der Vollständigkeit der Daten und zur Sicherstellung der Reihenfolge.

In dem Zusammenhang mit IPv6 über IPv4 wurde bereits ein Tunnel Broker Modell (siehe RFC3053 und Kapitel 6.3.2) definiert, in dem es einem Client ermöglicht wird, einen Tunnel über sein Interface zu nützen. Es gab bis zum TSP kein Protokoll, welches ein Aushandeln von "Parameters Of Time" ermöglichte. Dieser Missstand wurde durch das TSP behoben, da durch den Internet-Draft das vorhandene Tunnel Broker Modell erweitert wurde. Es ist dadurch ein *Control Protocol* vorhanden, das zwischen dem Broker und dem Client ausgetauscht wird, und das Aushandeln von Parametern zwischen zwei Teilnehmern durchführbar macht. Beispiel: Fordert ein Client ein Merkmal an, das vom Provider nicht angeboten wird, kann der Client fortfahren, ohne dieses Feature einzusetzen. Weiterhin kann der Server dem Client eine Link-Liste zusenden, die Server enthält, die das gewünschte Merkmal für den Client anbieten kann. Diese Eigenschaft steht ohne TSP nicht zur Verfügung.

Das Tunnel Setup Protocol bedient sich der *Authentication Phase* und der *Command Phase*.
Die Authentication Phase findet statt, wenn der Tunnel Broker oder Server seine Fähigkeiten den

Zeile	Befehl	Erläuterung
1	–Successful TCP Connection–	–Aufbau der Sitzung–
2	C: VERSION=1.0 CR LF	Senden der Versionsnummer
3	S: 1302 unsupported client version CR LF	Fehlermeldung durch den Server
4	<tunnel action="list" type="broker">	* Beginn Serverliste *
5	<broker>	
6	<address type="ipv4">1.2.3.4</address>	Serveradresse im IPv4-Format
7	</broker>	
8	<broker>	
9	<address type="dn">ts1.isp1.com</address>	Serveradresse als DN
10	</broker>	
11	</tunnel>	* Ende Serverliste *
12	–Connection Closed–	–Abbau der Sitzung–

Abbildung 7.1: Beispiel: Version nicht unterstützt mit Serverliste *[BDC01]*

Tunnel Clients anbietet und wenn sich die Tunnel Clients beim Server authentisieren. Fragen Clients einen Tunnel an oder werden die Daten aktualisiert, ist die Command Phase aktiv.

Die *Authentication Phase* besteht aus 3 Schritten: 1.) Die Identifikation der verwendeten Protokollversion des Clients. 2.) Das Anbieten der Serverfähigkeiten und Ressourcen. 3.) Die Client-Authentisierung.
Beim Beginn einer TCP-Sitzung mit einem Tunnel Server sendet der Tunnel Client seine aktuelle Protokollversionsnummer zum Server. Unterstützt der Server diese Version nicht, sendet er eine Fehlermeldung und schließt die Sitzung. Optional kann dem Client eine Liste mit erreichbaren kompatiblen Servern geschickt werden, wie es in der Abbildung 7.1 verdeutlicht wird. Akzeptiert der Server die Version sendet der Server eine Anfrage mit allen Informationen, die für die Authentisierung und die Konfiguration des Tunnel nötig sind (s. Abb 7.2 Zeile 3). Hierzu gehören die Art des Tunnels, das Authentisierungsschema und der Benutzername. Der Client entscheidet, ob die angebotene Tunnelart ausreichend ist. Er kann die Sitzung beenden oder die Authentisierung fortführen. Entscheidet der Client sich für den Server, erfolgt die Authentisierung des Clients beim Server (s. Abb 7.2 Zeile 4). In jedem Fall sendet der Server eine Meldung über den Ausgang der Authentisierung an den Tunnel-Client. Schlägt die Anmeldung fehl, erhält die Zeile 5 die Meldung "300 Authentication failed CR LF".

Nach einer erfolgreichen Anmeldung startet die *Command Phase*, in der von dem Tunnel Client eine Tunnelanfrage oder eine Tunnelaktualisierung an den Server geschickt wird. Auch hier werden die Nachrichten im XML-Format ausgetauscht. In der ersten Zeile steht die „Content-length", die angibt wie groß die folgende XML-Nachricht ist, wobei nur Zeichen gezählt werden. Dadurch ist dem Tunnelserver bekannt, wann die gesamte Nachricht angekommen und durchgearbeitet ist. Die Serverantwort enthält ebenfalls in der ersten Zeile die Größenangabe und in der zweiten einen "Return Code", um die Antwort später zuordnen zu können. Die XML-Message folgt in der

Zeile	Befehl/Beschreibung
1	–Successful TCP Connection–
2	Client: VERSION=1.0 CR LF
3	Server: CAPABILITY TUNNEL=V6V4 AUTH=DIGEST-MD5 CR LF
4	Client: AUTHENTICATE ANONYMOUS CR LF
5	Server: 200 Authentication successful CR LF

Abbildung 7.2: Beispiel: Authentisierung erfolgreich *[BDC01]*

dritten Zeile. Die Größenangabe bei der Antwort bezieht sich auf alle nachfolgenden Zeichen, einschließlich der Return Code-Zeile.

Topology

Die Abbildungen 7.3 und 7.4 beschreiben typische TSP Abläufe, um einen konfigurierten Tunnel zu erhalten. Man vergleiche hierzu die Abbildungen 6.6 und 6.7 und das Kapitel 6.3.2.

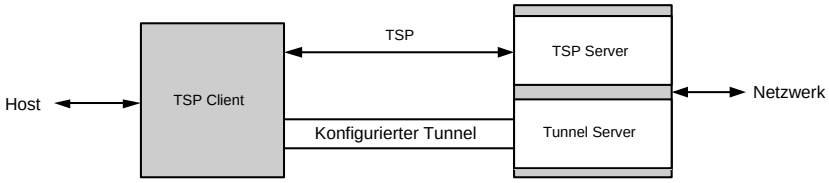


Abbildung 7.3: TSP beim Tunnel Server Modell *[BDC01]*

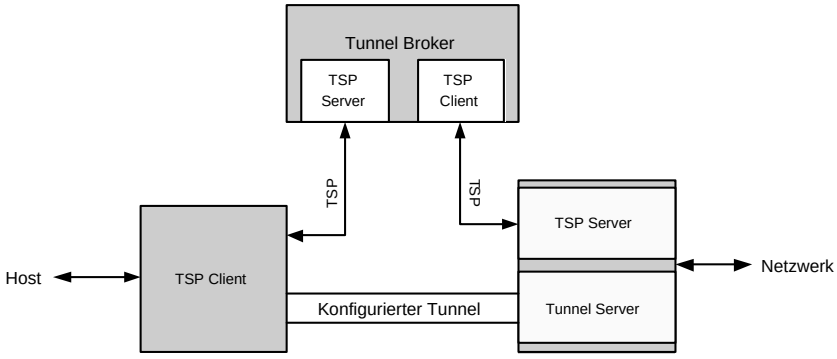


Abbildung 7.4: TSP beim Tunnel Broker Modell *[BDC01]*

Kapitel 8

Auswertung und Zusammenfassung

Die fortschreitenden Entwicklungen in technischen Bereichen und ansteigenden Internetnutzerzahlen erfordern eine Weiterentwicklung des bestehenden Internetprotokolls IPv4. Nach der Präsentation von drei Vorschlägen im Jahre 1992 wurde die Entscheidung durch das IPng-Direktorium für ein auf SIPP basierenden Protokoll getroffen. Die Anzahl der benötigten Adressen nimmt weiter zu, so dass durch die Einführung des CDIR bei IPv4 dieser Engpass entschärft wurde. Auf lange Sicht ist dieses Verfahren jedoch nicht ausreichend und das neue Protokoll erreicht durch die Einführung von 128Bit-Adressen eine Vervielfachung des zur Verfügung stehenden Adressraumes. Durch Umstrukturierung und Einteilung des Adressraumes in Multicast- und Unicast-Adressen wird ein erheblicher Geschwindigkeitsgewinn bei der Verarbeitung der Datenpakete erreicht. Der Umstieg von klassenbasierten Adressbereichen in hierarchische Strukturen ermöglicht eine Verringerung der benötigten Routingtabellen und deren Größe. Weiterhin wird die Adressierung flexibler und bietet mehr Ressourcen.

Die Effizienz von IPv6 steigt gegenüber IPv4 an, da durch den neuen Headeraufbau eine redundanzfreie Übertragung möglich ist. Die Berechnung der Checksumme bei IPv4 ist bei IPv6 nicht mehr vorhanden, was zur Effizienzsteigerung der Router beiträgt. Es werden nicht mehr alle Informationen in einen Header geschrieben und dem Paket vorangestellt, sondern es gibt einen Basisheader und mehrere optionale Erweiterungsheader, die unterschiedliche Aufgaben erfüllen. Da die Erweiterungsheader optional sind, werden ausschließlich die Erweiterungsheader verwendet, die für das jeweilige Paket benötigt werden. Die Festlegung einer Headerreihenfolge bewirkt eine zusätzliche Verkürzung der Verarbeitungsdauer. Die Router werten nicht mehr jeden Header aus, sondern beenden die Verarbeitung sobald ein nicht relevanter Erweiterungsheader vorliegt. Messungen mit *ping* und *ping6* von dem eigenen Server-Rechner ergaben folgende Werte:

```
ns1:/etc # ping -c 5 139.30.208.204
--- 139.30.208.204 ping statistics ---
5 packets transmitted, 5 received, 0% loss, time 3996ms
rtt min/avg/max/mdev = 0.100/0.197/0.583/0.193 ms
```

```
ns1:/etc # ping6 -c 5 2001:638:804:8000::204
--- 2001:638:804:8000::204 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max = 0.093/0.094/0.098 ms
```

Es ist zu erkennen, dass die Zeit für einen *round-trip* beim IPv6-ping kürzer ist, was auf den optimierten Header zurückzuführen ist. Selbst bei der Ausführunng des ping(6) auf dem eigenen System ist eine Geschwindigkeitserhöhung zu verzeichnen, was bei einer weltweiten Ausdehnung des IPv6-Netzes zu einer erheblichen Verringerung der Übertragungszeit führt.

Zusätzlich verzeichnet IPv6 einen Sicherheitsgewinn durch die Einführung der Authentisierungs- und Verschlüsselungsheader, was durch zusätzliche Bits bewerkstelligt wird, welche allerdings die Übertragungsrate verringern. Dennoch kann dem Faktor Sicherheit bei IPv6 eine Verbesserung gegenüber IPv4 zugesprochen werden, da die Mechanismen zur Authentisierung und Verschlüsselung leichter zu implementieren sind und dennoch einen besseren Schutz bieten.

Die Neugestaltung des Internet Control Message Protocols und die dadurch erreichte Plug-and-Play-Fähigkeit von IPv6 stellt einen wichtigen Vorteil dar. Die automatische Adresskonfiguration erleichtert Administratoren die Arbeit erheblich, was bei steigender Anzahl an Rechnern eine wichtige Rolle spielt, da die Netze nicht mehr zu überblicken sind. Menschen werden, auch aus beruflichen Gründen, mobiler und somit wird ebenfalls der Arbeitsplatz mobil. Aufgrund dessen und der Fortschritte im Mobilfunkbereich ist handlungsbedarf bezüglich der nötig. Die bessere Echtzeitfähigkeit von IPv6 und die mobilen Optionen ermöglichen eine schnelle und mobile Kommunikation von jedem Standpunkt der Welt aus mit der gleichen IPv6-Adresse.

Die momentane Verfügbarkeit von IPv6-Netzen ist nicht besonders groß, so dass viele Nutzer ein Tunnelverfahren anwenden müssen. Weiterhin sind nicht viele Internetseiten mit IPv6-Connectivity ausgestattet, wodurch der vorliegende Nutzen wesentlich verbessert werden muss. Mit den verfügbaren Internetseiten verlief ein Datenaustausch über IPv6 sehr zufriedenstellend.

Die Installation des Linux Client verlief problemlos, dank der einfach strukturierten Skripte des Tunnelbrokers freenet6 und der Tatsache, dass nur ein Browser und ein Tunnel implementiert werden mussten. Beim Einrichten des Linux Servers entstanden mehrere Probleme. Hauptsächlich im Bereich der Anpassung an das IPv6-Protokoll. Die Dokumentationen sind zwar sehr umfangreich, aber es werden die Möglichkeiten und Anweisungen für eine IPv6-Installation sehr lückenhaft dargestellt, wodurch sich der Arbeitsaufwand drastisch erhöht. Der Apache-Server ist verhältnismäßig schnell installiert und einsatzbereit, wohingegen der FTP-Server und der DNS mit viel Aufwand verbunden sind. Nach Angabe der Dokumentation zu PureFTPD ist es möglich, Usern den Zugang nur über SSH zu erlauben. Dieses Eigenschaft konnte nicht eingerichtet werden.

Die Internetdomains für den Server lauten:

```
IPv4 www http://www.ipv4.comlab.uni-rostock.de
IPv4 ftp ftp://ftp.ipv4.comlab.uni-rostock.de
IPv6 www http://www.ipv6.comlab.uni-rostock.de
IPv6 www http://[2001:638:804:8000::204]
IPv6 ftp ftp://ftp.ipv6.comlab.uni-rostock.de
```

Der Aufbau einer vollständigen IPv6-Internetseite, die einen Überblick über das Projekt gibt wird im Rahmen der Tätigkeiten einer studentischen Hilfskraft durchgeführt.

Fazit Die Neuerungen von IPv6 fallen bei einem Home-Anschluss nicht sonderlich ins Gewicht. Dies ändert sich, sobald das IPv6-Netz ausgebaut ist und sich die Nutzerzahl stark erhöht. Dadurch steigen die Anforderungen und Auslastungen der Netze an, was sich auf die Übertragungsgeschwindigkeit auswirkt. Erst dann werden sich die Veränderungen bei IPv6 auswirken. Das Internetprotokoll IPv6 ist ein effektives und zukunftsorientiertes Protokoll, das Möglichkeiten für weitere Erweiterungen besitzt. Jedoch ist das Netz zum jetzigen Zeitpunkt nicht besonders groß und die verfügbaren Anwendungen und Webseiten sind rar, so dass der Einsatz von IPv6 in modernen Kommunikationswegen hauptsächlich für Forschungs- und Testzwecke sinnvoll ist.

Anhang A

Konfigurationsdatei Tunnelbroker

A.1 freenet6 tspc.conf

```
#----------------------------------------------------------------
# tspc.conf
#----------------------------------------------------------------

#
# tsp client version
#
tsp_version=1.0.0
tsp_dir=/usr/local/freenet6

#
# auth_method=any|supported mechanism
# if any is specified, the first mechanism that is supported by
# both the server and the client will be picked. The order of
# precedence is themechnism list shown by the output of -h.
# supported mechanism is one of the token shown by th output
# of -h (case sensitive).
auth_method=any

#
# client_v4=auto|1.1.1.1 (valid ip address)
# if auto is specified, if_source or -s cmd line option must be
# specified.
client_v4=auto
```

```
# userid=anonymous
# the userid can be anonymous or any alphanumeric value that is
# dns legal.
userid=comlab2-uni-rostock

# passwd=
# The passwd must be empty if userid is anonymous or an
# alphanumeric string
# if userid is not anonymous.
passwd=******* #GESCHÜTZT

# template=target
# template tells which configuration script needs to be run for
# interface setup.
# Normally you only need to put your osname in the value.
# You can also use "checktunnel" if you only want the tunnel
# info print out.
#template=checktunnel
template=linux

server=tsps1.freenet6.net
#server=localhost



if_tunnel=sit0
# end of tspc.conf
#------------------------------------------------------------
```

Anhang B

Konfigurationsdateien Apache

B.1 Apache Server

httpd.conf Auszug aus der Konfifurationsdatei des Apache-Servers.

```
#
# Based upon the NCSA server configuration files originally by
# Rob McCool.
# This is the main Apache server configuration file. It contains
# the configuration directives that give the server its
# instructions. See <URL:http://httpd.apache.org/docs-2.0/> for
# detailed information about
# the directives.
#
# Do NOT simply read the instructions in here without
# understanding what they do.  They're here only as hints or
# reminders.  If you are unsure consult the online docs. You
# have been warned.
# Configuration directives are grouped into three basic
# sections:
#  1. Directives that control the operation of the Apache server
#     process as a whole (the 'global environment').
#  2. Directives that define the parameters of the 'main' or
#    'default' server, which responds to requests that aren't
#     handled by a virtual host. These directives also provide
#     default values for the settings of all virtual hosts.
#  3. Settings for virtual hosts, which allow Web requests to be
#     sent to different IP addresses or hostnames and have them
```

```
#     handled by the same Apache server process.
#
# Configuration and logfile names: If the filenames you specify
# for many of the server's control files begin with "/" (or
# "drive:/" for Win32), the server will use that explicit path.
# If the filenames do *not* begin with "/", the value of
# ServerRoot is prepended -- so "logs/foo.log" with ServerRoot
# set to "/usr/local/apache2" will be interpreted by the
# server as "/usr/local/apache2/logs/foo.log".
#

### Section 1: Global Environment
#
# The directives in this section affect the overall operation of
# Apache, such as the number of concurrent requests it can
# handle or where it can find its configuration files.
# ServerRoot: The top of the directory tree under which the
# server's configuration, error, and log files are kept.
ServerRoot "/usr/local/apache2"

# Timeout: The number of seconds before receives and sends time
# out.
Timeout 300

# KeepAlive: Whether or not to allow persistent connections
# (more than one request per connection). Set to "Off" to
# deactivate.
KeepAlive On

# MaxKeepAliveRequests: The maximum number of requests to allow
# during a persistent connection. Set to 0 to allow an unlimited
# amount. We recommend you leave this number high, for maximum
# performance.
MaxKeepAliveRequests 100

# KeepAliveTimeout: Number of seconds to wait for the next
# request from the same client on the same connection.
KeepAliveTimeout 15

# Listen: Allows you to bind Apache to specific IP addresses
# and/or ports, in addition to the default. See also the
# <VirtualHost> directive.
#Listen 12.34.56.78:80
```

```
#Listen 80
Listen [::]:80

# Dynamic Shared Object (DSO) Support
# To be able to use the functionality of a module which was
# built as a DSO youhave to place corresponding ‘LoadModule’
# lines at this location so the directives contained in it are
# actually available _before_ they are used. Statically compiled
# modules (those listed by ‘httpd -l’) do not need to be loaded
# here. Example:
# LoadModule foo_module modules/mod_foo.so
LoadModule php4_module modules/libphp4.so


### Section 2: ’Main’ server configuration
# The directives in this section set up the values used by the
# ’main’ server, which responds to any requests that aren’t
# handled by a <VirtualHost> definition.  These values also
# provide defaults for any <VirtualHost> containers you may
# define later in the file.
# All of these directives may appear inside <VirtualHost>
# containers, in which case these default settings will be
# overridden for the virtual host being defined.
<IfModule !mpm_winnt.c>
<IfModule !mpm_netware.c>
#
# If you wish httpd to run as a different user or group, you
# must run httpd as root initially and it will switch.
#
User nobody
Group #-1
</IfModule>
</IfModule>
#
# ServerAdmin: Your address, where problems with the server
# should  be e-mailed.  This address appears on some server-
# generated pages, such as error documents.
# e.g. admin@your-domain.com
ServerAdmin staudte@comlab.uni-rostock.de

# UseCanonicalName: Determines how Apache constructs self-
# referencing URLs and the SERVER_NAME and SERVER_PORT
# variables. When set "Off", Apache will use the Hostname and
```

```
# Port supplied by the client.  When set "On", Apache will use
# the value of the ServerName directive.
#
UseCanonicalName Off

# DocumentRoot: The directory out of which you will serve your
# documents. By default, all requests are taken from this
# directory, but symbolic links and aliases may be used to point
# to other locations.
DocumentRoot "/usr/local/apache2/www"

#
# This should be changed to whatever you set DocumentRoot to.
#
<Directory "/usr/local/apache2/www">

# DirectoryIndex: sets the file that Apache will serve if a
# directory is requested.
# The index.html.var file (a type-map) is used to deliver
# content-negotiated documents.  The MultiViews Option can be
# used for the same purpose, but it is much slower.
#
DirectoryIndex index.php index.html index.html.var index.htm
index.php index.php3

# AccessFileName: The name of the file to look for in each
# directory for access control information.  See also the
# AllowOverride directive.
#
AccessFileName .htaccess

#
# This should be changed to the ServerRoot/manual/.  The alias
# provides the manual, even if you choose to move your
# DocumentRoot.  You may comment this out if you do not care for
# the documentation.
#
Alias /manual "/usr/local/apache2/manual"

<Directory "/usr/local/apache2/manual">
    Options Indexes FollowSymLinks MultiViews
    AllowOverride None
    Order allow,deny
```

```
    Allow from all
</Directory>

# ScriptAlias: This controls which directories contain server
# scripts. ScriptAliases are essentially the same as Aliases,
# except that documents in the realname directory are treated as
# applications and run by the server when requested rather than
# as documents sent to the client. The same rules about trailing
# "/" apply to ScriptAlias directives as to Alias.
#
ScriptAlias /cgi-bin/ "/usr/local/apache2/cgi-bin/"

<IfModule mod_cgid.c>
#
# Additional to mod_cgid.c settings, mod_cgid has Scriptsock
# <path> for setting UNIX socket for communicating with cgid.
#
#Scriptsock            logs/cgisock
</IfModule>

# "/usr/local/apache2/cgi-bin" should be changed to whatever
# your ScriptAliased CGI directory exists, if you have that
# configured.
#
<Directory "/usr/local/apache2/cgi-bin">
    AllowOverride None
    Options None
    Order allow,deny
    Allow from all
</Directory>

# AddEncoding allows you to have certain browsers (Mosaic/X
#  2.1+) uncompress information on the fly. Note: Not all
# browsers support this. Despite the name similarity, the
# following Add* directives have nothing to do with the
# FancyIndexing customization directives above.
#
AddEncoding x-compress Z
AddEncoding x-gzip gz tgz

# AddType allows you to add to or override the MIME
# configuration file mime.types for specific file types.
#
```

```
AddType application/x-tar .tgz
AddType application/x-httpd-php .php


### Section 3: Virtual Hosts
#
# VirtualHost: If you want to maintain multiple
# domains/hostnames on your machine you can setup VirtualHost
# containers for them. Most configurations use only name-based
# virtual hosts so the server doesn't need to worry about
# IP addresses. This is indicated by the asterisks in the
# directives below.
# Please see the documentation at
# <URL:http://httpd.apache.org/docs-2.0/vhosts/>
# for further details before you try to setup virtual hosts.
# You may use the command line option '-S' to verify your
# virtual hostconfiguration.
```

Anhang C

Konfigurationsdateien DNS

C.1 /etc/nsswitch.conf

```
#
# /etc/nsswitch.conf
#
# An example Name Service Switch config file. This file should
# be sorted with the most-used services at the beginning.
# Legal entries are:
# compat Use Libc5 compatibility setup
# nisplus Use NIS+ (NIS version 3)
# nis Use NIS (NIS version 2), also called YP
# dns Use DNS (Domain Name Service) for IPv4 only
# dns6 Use DNS for IPv4 and IPv6
# files Use the local files
# db Use the /var/db databases
# [NOTFOUND=return] Stop searching if not found so far
#
# For more information, please read the nsswitch.conf.5
# manual page.
passwd: compat
group: compat

#HIER MUSS dn6 ANSTELLE VON dns eingetragen werden!
hosts:    files dns6
networks: files dns6

services: files
```

```
protocols: files
rpc: files
ethers: files
netmasks: files
netgroup: files
publickey: files
bootparams: files
automount: files
aliases: files
```

C.2 etc/hosts.conf

```
#
# /etc/host.conf - resolver configuration file
#
# Please read the manual page host.conf(5) for more
# information.
#
# The following option is only used by binaries linked
# against libc4 or libc5. This line should be in sync
# with the "hosts" option in /etc/nsswitch.conf.
#
#HIER STEHT DIE REIHENFOLGE
order hosts, bind
#
# The following options are used by the resolver library:
#
multi on
```

C.3 /etc/hosts

```
#
# hosts
# This file describes a number of hostname-to-address
# mappings for the TCP/IP subsystem.  It is mostly
# used at boot time, when no name servers are running.
# On small systems, this file can be used instead of a
# "named" name server.
# Syntax:
#
# IP-Address  Full-Qualified-Hostname  Short-Hostname
#
127.0.0.1   localhost
127.0.0.2   localhost
# special IPv6 addresses
::1       localhost
fe00::0     ipv6-localnet
ff02::1     ipv6-mcastprefix
ff02::2     ipv6-allnodes
ff02::3     ipv6-allhosts
```

C.4 /etc/resolv.conf

```
### BEGIN INFO
#
# Modified_by:  dhcpcd
# Backup:       /etc/resolv.conf.saved.by.dhcpcd
# Process:      dhcpcd
# Process_id:   2887
# Script:       /sbin/modify_resolvconf
# Saveto:
# Info:
#    This is a temporary resolv.conf created by
#    dhcpcd.  A previous resolv.conf has been saved as
#    /etc/resolv.conf.saved.by.dhcpcd and will be restored
#    when dhcpcd is stopped for any reason.  If you don't
#    like dhcpcd to change your nameserver settings
#    then either set DHCLIENT_MODIFY_RESOLV_CONF=no
#    in /etc/rc.config.d/dhcpcd.rc.config, or set
#    MODIFY_RESOLV_CONF_DYNAMICALLY=no in /etc/rc.config or
#    (manually) use dhcpcd with -R.  If you only want to keep
#    your searchlist, set DHCLIENT_KEEP_SEARCHLIST=yes in
#    /etc/rc.config.d/dhcpcd.rc.config or (manually) use the
#    -S option.
#
### END INFO

domain comlab.uni-rostock.de
search comlab.uni-rostock.de
nameserver 139.30.208.204
nameserver 139.30.208.2
nameserver 139.30.8.7
```

C.5 /etc/named.conf

```
#------------------------------------------------------------
# named.conf
#------------------------------------------------------------
#
#
# Konfigurationsdatei für DNS Bind
# geschrieben von
# Tobias Staudte
# staudte@comlab.uni-rostock.de
#
#
#------------------------------------------------------------
#------------------------------------------------------------
#Definition von globalen Optionen
options{
# Das Verzeichnis für die Zonendateien
     directory "/var/named";
# pid-file
pid-file "named.pid";
#Anweisung an named, dass er nur als Weiterleiter arbeitet
forward only;
#Liste der externen DNS-Server
forwarders{
139.30.8.7;
139.30.8.8;
127.0.0.1;
};
listen-on-v6 { any; };
# Sollen rekursive Anfragen an andere DNS-Server erlaubt sein,
     # wenn unser Server eine Anfrage nicht beantworten kann
     # recursion yes;
allow-transfer {139.30.8.7; 139.30.8.8; 139.30.208.204;};
};
#------------------------------------------------------------
#Definition der Zonen
#Zonen für den localhost und der umgekehrten Namensauflösung
  zone "localhost" IN {
    type master;
    file "localhost.zone";
  };
```

```
  zone "0.0.127.in-addr.arpa" IN {
    type master;
    file "127.0.0.zone";
  };
zone "1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.\
0.0.0.0.ip6.int" in {
type master;
file "::1.zone";
  };

#Damit der Nameserver den Kontakt zu allen anderen
#Nameservern im Internet aufnehmen kann definieren wir jetzt
#eine allgemeine Zone. "root.hint" steht in dem oben angegebenen
#Verzeichnis
zone "." IN {
    type hint;
    file "root.hint";
  };

#Zonen für das Netz bzw den Server
zone "ipv6.comlab.uni-rostock.de" in {
type master;
file "ipv6.zone";
};
zone "ipv4.comlab.uni-rostock.de" in {
type master;
file "ipv4.zone";
};
zone "208.30.139.in-addr.arpa" in {
type master;
file "139.30.208.zone";
};
# Die nachfolgende Reverse-Zone steht in einer Zeile.
# vgl. IPv4-Reverse-Zonen
zone "4.0.2.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.8.4.0.8.0.8.3.6.0.\
1.0.0.2.ip6.int" IN {
type master;
file "2001:638:804:8000::204.zone";
};
```

C.6 Zonendateien

C.6.1 Zonendatei 127.0.0.zone

```
;-------------------------------------------------------------
;Gültigkeitsdauer (Time To Live). Hier 1 Tag.
;-------------------------------------------------------------
;Änderungen werden also spätestens nach einem Tag aktualisiert,
;wenn er nicht vorher neu gestartet wurde
;(/etc/init.d/named restart).
$TTL 1D
$ORIGIN .
;-------------------------------------------------------------
;Start of Authority SOA
;-------------------------------------------------------------
;Domainnamen. IN SOA Nameservername Email_des_Verwalters (
;                                   Seriennummer
;                                   Refreshwert
;                                   Wiederholungsintervall
;                                   Ablaufzeit
;                                   Minimum )
;WICHTIG ist die Seriennummer
;(JJJJMMTT+angehängte 2-stellige Zahl).
;Nameserver finden so heraus, wer den neuesten Eintrag besitzt.
0.0.127.in-addr.arpa IN SOA ns1.ipv4.comlab.uni-rostock.de. \
                            2003011303;
                          8H;
                          2H;
                          1W;
                          1D)
;-------------------------------------------------------------
;Nameserver und Mailserver
;-------------------------------------------------------------
;Der erste Nameserver (NS) der Domain heißt ipv4.
;Ein Mailserver wäre mit MX (Mail Exchange) angegeben worden
        IN  NS  ns1.ipv4.comlab.uni-rostock.de.
        IN NS nagilum.comlab.uni-rostock.de.
        IN NS hp1.uni-rostock.de.
        IN NS top.uni-rostock.de.
;-------------------------------------------------------------
;Angabe aller an die Domain angeschlossenen Rechner
;-------------------------------------------------------------
```

```
$ORIGIN 0.0.127.in-addr.arpa.
1 IN PTR localhost.
2 IN PTR localhost.
;$ORIGIN 0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.ip6.int.
;1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0 IN PTR localhost.
```

C.6.2 Zonendatei localhost.zone

```
$TTL 1D
localhost. IN SOA ns1.ipv4.comlab.uni-rostock.de.\
                  hostmaster.comlab.uni-rostock.de (
                         2003011302;
                          8H;
                        2H;
                        1W;
                        1D)
;-------------------------------------------------------
;Nameserver und Mailserver
;-------------------------------------------------------
        IN  NS  ns1.ipv4.comlab.uni-rostock.de.
        IN NS nagilum.comlab.uni-rostock.de.
        IN NS hp1.uni-rostock.de.
        IN NS top.uni-rostock.de.
;-------------------------------------------------------
;Angabe aller an die Domain angeschlossenen Rechner
;-------------------------------------------------------
IN A 127.0.0.1
IN A 127.0.0.2
IN AAAA ::1
```

C.6.3 Zonendatei 208.30.139.zone

```
;Zonendatei, um die IP-Adresse dem Namen zuzuteilen
;file: /var/named/208.30.139.zone
$TTL 1D
$ORIGIN .
;-------------------------------------------------------
;Start of Authority SOA
;-------------------------------------------------------
```

```
208.30.139.in-addr.arpa IN SOA  ns1.ipv4.comlab.uni-rostock.de.
                                2003011303      ; serial
                                1D              ; refresh
                                2H              ; retry
                                1W              ; expiry
                                2D )            ; minimum

;-------------------------------------------------------
;Nameserver und Mailserver
;-------------------------------------------------------
        IN  NS  ns1.ipv4.comlab.uni-rostock.de.
        IN NS nagilum.comlab.uni-rostock.de.
        IN NS hp1.uni-rostock.de.
        IN NS top.uni-rostock.de.

;-------------------------------------------------------
;Angabe aller an die Domain angeschlossenen Rechner
;-------------------------------------------------------
$ORIGIN 208.30.139.in-addr.arpa.
204  PTR  ns1.ipv4.comlab.uni-rostock.de.
```

C.6.4 Zonendatei ipv4.zone

```
;Zonendatei
;file: /var/named/ipv4.zone
$TTL 1D
$ORIGIN comlab.uni-rostock.de.
;-------------------------------------------------------
;Start of Authority SOA
;-------------------------------------------------------
ipv4 IN SOA ns1.ipv4.comlab.uni-rostock.de.
                        hostmaster.comlab.uni-rostock.de. (
                                2003012101    ; serial
                                1D              ; refresh
                                2H              ; retry
                                1W              ; expiry
                                2D )            ; minimum

;-------------------------------------------------------
;Nameserver und Mailserver
```

```
;-------------------------------------------------------
        IN  NS  ns1.ipv4.comlab.uni-rostock.de.
        IN NS nagilum.comlab.uni-rostock.de.
        IN NS hp1.uni-rostock.de.
        IN NS top.uni-rostock.de.

;-------------------------------------------------------
;Angabe aller an die Domain angeschlossenen Rechner
;-------------------------------------------------------
$ORIGIN ipv4.comlab.uni-rostock.de.
ns1 IN A     139.30.208.204
www IN  CNAME ns1.ipv4.comlab.uni-rostock.de.
ftp IN CNAME ns1.ipv4.comlab.uni-rostock.de.
```

C.6.5 Zonendatei 2001:638:804:8000::204.zone

```
;Zonendatei
;file: /var/named/2001:638:804:8000::204.zone
$TTL 86400
;Die nachfolgenden 2 Zeilen stehen in einer Zeile
$ORIGIN 4.0.2.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.8.4.0.8.0.\
        8.3.6.0.1.0.0.2.ip6.int.
@ IN SOA ns1.ipv4.comlab.uni-rostock.de.
         hostmaster.comlab.uni-rostock.de. (
         2003012101 ; serial
        21600      ; refresh (6 hours)
        3600       ; retry (1 hour)
        604800     ; expire (1 week)
        172800     ; minimum (2 days)
        )
;-------------------------------------------------------
;Nameserver und Mailserver
;-------------------------------------------------------
        IN NS ns1.ipv4.comlab.uni-rostock.de.
        IN NS nagilum.comlab.uni-rostock.de
        IN NS ns.join.uni-muenster.de.
        IN NS ns6.dns.space.net.
        IN NS ws2.nl.ten-155.net.
        IN NS ws-lei1.win-ip.dfn.de.
;-------------------------------------------------------
;Angabe aller an die Domain angeschlossenen Rechner
```

```
;-------------------------------------------------------------
;Die nachfolgenden 2 Zeilen stehen in einer Zeile
$ORIGIN 4.0.2.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.8.4.0.8.0.\
        8.3.6.0.1.0.0.2.ip6.int.
  IN PTR www.ipv6.comlab.uni-rostock.de.
```

C.6.6 Zonendatei ipv6.zone

```
;Zonendatei
;file: /var/named/ipv6.zone

;-------------------------------------------------------------
;Gültigkeitsdauer (Time To Live). Hier 1 Tag.
;-------------------------------------------------------------
$TTL 1D
$ORIGIN comlab.uni-rostock.de.
;-------------------------------------------------------------
;Start of Authority SOA
;-------------------------------------------------------------
ipv6 IN SOA ns1.ipv4.comlab.uni-rostock.de.
            hostmaster.comlab.uni-rostock.de. (
                              2003012102    ; serial
                              1D              ; refresh
                              2H              ; retry
                              1W              ; expiry
                              2D )            ; minimum

;-------------------------------------------------------------
;Nameserver und Mailserver
;-------------------------------------------------------------
        IN  NS  ns1.ipv4.comlab.uni-rostock.de.
        IN NS nagilum.comlab.uni-rostock.de.
        IN NS hp1.uni-rostock.de.
        IN NS top.uni-rostock.de.

;-------------------------------------------------------------
;Angabe aller an die Domain angeschlossenen Rechner
;-------------------------------------------------------------
$ORIGIN ipv6.comlab.uni-rostock.de.
www IN AAAA  2001:638:804:8000::204
```

```
www   IN A    139.30.208.204
ftp   IN AAAA  2001:638:804:8000::204
```

C.6.7 Zonendatei ::1.zone

```
$TTL 3D
;Die nachfolgenden 2 Zeilen stehen in einer Zeile
$origin 1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.\
        0.0.0.0.0.0.0.0.ip6.int.
@ IN SOA ns1.ipv4.comlab.uni-rostock.de.
                  hostmaster.comlab.uni-rostock.de (
                 2003011301;
                 8H;
                 2H;
                 1W;
                 1D)
;-------------------------------------------------------------
;Nameserver und Mailserver
;-------------------------------------------------------------
        IN  NS  ns1.ipv4.comlab.uni-rostock.de.
        IN NS nagilum.comlab.uni-rostock.de.
        IN NS hp1.uni-rostock.de.
        IN NS top.uni-rostock.de.

;-------------------------------------------------------------
;Angabe aller an die Domain angeschlossenen Rechner
;-------------------------------------------------------------
;$ORIGIN 1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.\
         0.0.0.0.0.0.0.0.ip6.int.
IN PTR localhost.
```

Anhang D

Konfigurationsdatei boot.local

```
#! /bin/sh
#
# Copyright (c) 1996 SuSE GmbH Nuernberg, Germany.
# All rights reserved.
# Author: Werner Fink <werner@suse.de>, 1996
#         Burchard Steinbild <bs@suse.de>, 1996
#
# /etc/init.d/boot.local
#
# script with local commands to be executed from init on system startup
#
. /etc/rc.config

#
# Here you should add things, that should happen directly after booting
# before we’re going to the first run level.
#

#BSP18.12.2002: Added httpd
echo -n "Starting httpd from boot.local......"
/usr/local/apache2/bin/apachectl startssl
echo "done"

#DNS-Starten
echo -n "Starting Bind-DNS from boot.local......"
/usr/local/bind/sbin/named
echo "done"
```

Anhang E

Verzeichnisstrukturen

E.1 Client

```
Mozilla:
#Installationsverzeichnis
/usr/local/programme/mozilla/

IPv6-Tunnel über freenet6
#Internetseite:
www.freenet6.net
#Start
/usr/local/programme/freenet6/bin/
./tspc -vf ./tspc.conf
und
./tspc -vf ./tspc2.conf

FTP
# bereits in Linux integriert
```

E.2 Server

```
#Fileliste:
#skript: Verzeichnis Datei
Allgemeines:
#Bootdatei
/etc/init.d/boot.local
```

```
Mozilla:
#Installationsverzeichnis
/usr/local/mozilla/

SSL und SSH
#Openssl Installation
/usr/local/ssl/
#SSH-Installation
/usr/bin/
/usr/local/ssh

Apache-Server:
#start
/usr/local/apache2/bin/
./apachectl startssl
#stop
/usr/local/apache2/bin/
./apachectl stop
#Konfigurationsdatei
/usr/local/apache2/conf/httpd.conf
#Webseitenverzeichnis
/usr/local/apache2/www/

DNS:
#Bind9.2.1-Dateien
/usr/local/bind/
#Protokolladatei
/var/log/ message
#named-Datei
/etc/ named.conf
#Hostdatei
/etc/ hosts
#Reihenfolge der Rechner, die IP-Adresse einem
#Remoterechner zuweisen soll
/etc/ resolve.conf
#Zonendateien
/var/named/
#Starten
named

Pure-Ftpd:
#Pure-ftpd Installationspfad
```

```
/usr/local/pure-ftpd/
#Starten über inetd
inetd
#inetd-Konfigurationsfile
/etc/inetd.conf (letzte Zeile)
#Passwortfiles
/usr/local/pure-ftpd/etc -->pureftpd.passwd und pureftpd.pdb
#Dokumentationen
/usr/local/pure-ftpd/doku/
```

Abbildungsverzeichnis

1.1 Probleme von IPv4 7

1.2 Verbesserungen durch IPv6 7

2.1 Aufbau einer Multicast-Adresse *[RFC2373]* 9

2.2 Gültigkeitsbereiche einer Multicast-Adresse *[BH01]* 10

2.3 Einteilung des Adressraumes *[HU00] [RFC2373]* 10

3.1 Prinzip des Headeraufbaus 14

3.2 IPv6 Basis-Header *[RFC2460]* 14

3.3 Headertypen und Reihenfolge und Typen der Header *[BH01]* 15

3.4 IPv6 Routing-Header *[RFC2460]* 16

3.5 IPv6 Fragmentation Header *[RFC2460]* 17

3.6 IPv6 Hop-by-Hop-Header *[RFC2460]* 18

3.7 Authentication-Header *[RFC2402]* 20

3.8 Beispiel einer Nutzdaten-Verschlüsselung *[DI98]* 21

3.9 Tunnel-Verschlüsselung *[DI98]* 21

4.1 Aufbau des ICMP-Pakets *[RFC2463]* 22

4.2 ICMP-Typen *[RFC2463] [BH01]* 23

4.3 Caches des Neighbor Protocols *[BH01]* 27

4.4 Ablauf beim Absenden eines IPv6-Paketes im Sender *[BH01]* 27

4.5 DHCP-Advertise *[HU00]* 30
4.6 DHCP-Request *[HU00]* 30

5.1 Destination-Header Home Address *[DI98]* 32
5.2 Destination-Header Binding Update zum Anmelden *[DI98]* 33
5.3 Quittungsmeldungen Mobile-IPv6 *[IN03]* 35
5.4 Destination-Header Bindings-Abfrage *[DI98]* 35

6.1 Paraller Einsatz von IPv4 und IPv6 im LAN *[BH01]* 37
6.2 Dual-IP-Endsysteme am IPv4-Netz *[BH01]* 37
6.3 Kopplung von IPv6-Netzen über ein IPv4-Netz *[BH01]* 38
6.4 Kommunikation bei Erweiterung eines IPv4-Netzes *[BH01]* 38
6.5 IP-Header-Translation *[BH01]* 39
6.6 Tunnel-Broker-Modell *[RFC3053]* 41
6.7 Tunnel-Broker-Modell mit Dämon *[RFC3053]* 42

7.1 Beispiel: Version nicht unterstützt mit Serverliste *[BDC01]* 53
7.2 Beispiel: Authentisierung erfolgreich *[BDC01]* 54
7.3 TSP beim Tunnel Server Modell *[BDC01]* 54
7.4 TSP beim Tunnel Broker Modell *[BDC01]* 54

Literaturverzeichnis

[BH01] PROF. DR. ANATOL BADACH, DR. ERWIN HOFFMANN:
Technik der IP-Netze.
Carl Hanser Verlag München 2001

[HU00] CHRISTIAN HUITEMA:
IPv6 - die neue Generation.
Addison Wesley Verlag 2000.

[DI98] DIPL. INFORM. HANS PETER DITTLER:
IPv6 - das neue Internet Protokoll.
dpunkt Verlag 1998.

[BDC01] M. BLANCHET, R. DESMEULES, A CORMIER:
Internet Draft TSP.
Viagenie Inc. 2001.

[DO01] DIPL.-INF. RALF DÖRING:
Aufbruch in die neue Welt - IPv6 in IPv4 Netzen.
`www13.in.tum.de/personen/lachmund/papers/ipv6.pdf`

[RD01] AUTOR UNBEKANNT:
Der Weg von IPv4 zu IPv6.
`http://www.rdfnuernberg.de/iav1a/theorie/ipv4.html`

[NC02] AUTOR UNBEKANNT:
Migration auf das Internet-Protokoll Version 6.
Network Computing, Ausgabe 7-8/2002 vom 24.April 2002.

[LN00] SVEN LACHMUND UND MARKUS NOSKE:
Seminararbeit „IPv6-Unterschiede und Neuerungen gegenüber IPv4“.
Fachhochschule München Wintersemester 2000/2001.

[IN03] DIVERSE AUTOREN:
IPv6-NET.
`http://www.ipv6-net.org`

[MS03] MARTIN SCHMIDT:
X-Fish.
`http://www.x-fish.org`

[WPE99] JENS PRAHM, MICHAEL WIELSCH, HANS-GEORG ESSER:
Linux intern.
Data Becker GmbH Co. KG 1999.

[HA97] FRED HANTELMANN:
Linux für Durchstarter.
Springer Verlag Berlin Heidelberg 1997.

[IE02] DIVERSE AUTOREN:
Internetseite der Internet Engineering Task Force.
`http://www.ietf.org`

[go02] DIVERSE AUTOREN:
Suchmaschine google
`http://www.google.de`

[AB02] ERIK ABELE, RYAN BLOOM,RICH BOWEN, WEITERE:
Dokumentation Apache Webserver 2.0.
`http://httpd.apache.org/docs-2.0/`

[IS03] ISC - INTERNET SOFTWARE CONSORTIUM:
Dokumentation DNS Bind9.
`http://www.nominum.com/content/documents/bind9arm.pdf`

[GU00] ARNT GULBRANDSEN, JANOS FARKAS, WEITERE:
Dokumentation PureFTPd.
`http://www.pureftpd.org/documentation.shtml`