UNIVERSITÄT KOBLENZ · LANDAU
Fachbereich 4: Informatik

IPv6 Autokonfiguration

Studienarbeit

vorgelegt von

Christian Schwarz

Betreuer:	Dipl.-Inf. Harald Dickel, Institut für Informatik, Fachbereich Informatik
Erstgutachter:	Prof. Dr. Christoph Steigner, Institut für Informatik, Fachbereich Informatik
Zweitgutachter:	Dipl.-Inf. Harald Dickel, Institut für Informatik, Fachbereich Informatik

Koblenz, im Januar 2008

Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Mit der Einstellung dieser Arbeit in die Bibliothek bin ich einverstanden.
Der Veröffentlichung dieser Arbeit im Internet stimme ich zu.

Koblenz, den … Unterschrift

Übersicht

Diese Studienarbeit stellt verschiedene Möglichkeiten der automatischen Konfiguration von Netzwerkknoten unter IPv6, dem designierten Nachfolger des Internetprotokolls, vor. Dabei wird recht ausführlich in IPv6-Netzwerke eingeführt, wobei aber vorausgesetzt wird, dass der Leser Kenntnisse über das IP-Protokoll der Version 4 hat.

Es werden sowohl die zustandslose als auch DHCPv6 ausführlich in der Theorie als auch im praktischen Einsatz gezeigt. Dafür wird das VNUML-System eingesetzt, das an der Technischen Universität Madrid entwickelt wurde und das es ermöglicht, mehrere Linuxsysteme auf einem Wirtsrechner zu simulieren.

Inhaltsverzeichnis

Abkürzungsverzeichnis

- DAD : Duplicate Address Detection, Erkennung duplizierter Adressen
- DHCP: Dynamic Host Configuration Protocol
- DHCPv6: Dynamic Host Configuration Protocol Version 6
- DUID: DHCP Unique Identifier: eindeutiger DCHP-Bezeichner
- IA: Identity Association, Identitätszuordnung
- IAID: IA-Identifier, IA-Bezeichner
- IPv4: Internet Protokoll Version 4
- IPv6: Internet Protokoll Version 6
- VNUML: Virtual Network User Mode Linux

Abbildungsverzeichnis

Abbildungsverzeichnis

1 Einleitung

1.1 Problemstellung und Motivation

IPv6 ist der designierte Nachfolger des aktuell hauptsächlich im Internet verwendeten Protokolls IPv4. Dessen Adresspool wird in den nächsten Jahren vergeben werden – eine viel zitierte Studie [POTAROO] liefert Zahlen zwischen 2011 und 2020.

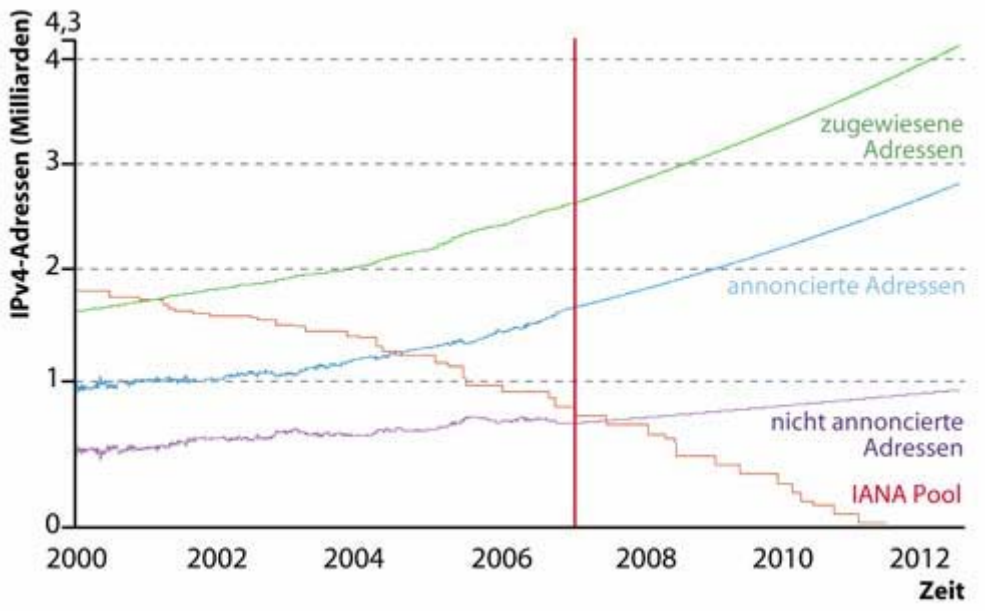

Abbildung 1.1 Prognostizierte Entwicklung der Vergabe der IPv4-Adressen laut [Heise]

Spätestens dann wird sich IPv6 durchsetzen, denn Alternativen sind derzeit keine in Sicht. Es ist also notwendig, sich besonders in der Lehre mit diesem neuen Protokoll zu beschäftigen. Ziel dieser Arbeit soll es dementsprechend sein, die automatische Konfiguration von Netzwerkknoten mit Adressen und sonstigen Konfigurationsparametern für die Lehre zu erörtern und ihren Einsatz in der Praxis zu demonstrieren. Dafür soll die in der Arbeitsgruppe von Professor Steigner bevorzugte Simulationsumgebung VNUML zum Einsatz kommen, die es ermöglicht, einfach Netzwerkszenarien zu konfigurieren und vollwertige Linuxrechner zu simulieren.

1.2 Aufbau der Arbeit

Die Arbeit ist wie folgt aufgebaut: Das folgende Kapitel wird zunächst in die Grundlagen zu IPv6 einführen. In Kapitel 3 wird dann das Verfahren der zustandslosen Autokonfiguration erläutert. Darauf folgt in Kapitel 4 die Darstellung der Funktionsweise von DHCPv6. In Kapitel 5 werden alternative Ansätze behandelt und in Kapitel 6 wird gezeigt, wie man die verschiedenen vorgestellten Verfahren in der Praxis einsetzt. Im abschließenden Kapitel 7 werden die Ergebnisse zusammengefasst.

2 IPv6 Grundlagen

2.1 Geschichte des Internet-Protokolls

Das Internetprotokoll wurde erstmalig Anfang 1980 in RFC 760 veröffentlicht und 1981 noch einmal in RFC 791 überarbeitet. Es erhielt die Versionsnummer 4, weil die Nummern 0 bis 3 Entwicklungsversionen des Protokolls zugewiesen worden waren. Dieses Protokoll im Rahmen dieser Arbeit als IPv4 bezeichnet um es begrifflich von IPv6 (Internet Protokoll Version 6) abzugrenzen.

Bereits zu Beginn der 1990er Jahre wurde klar, dass bei der rasanten Entwicklung des Internets bald der Adressraum von IPv4 ausgeschöpft sein würde. Diese Entwicklung wurde zwar durch *Classless Inter-Domain Routing* (RFC 1519) *und Network Adress Translation* (RFCs 1631 und 3022) abgemildert, allerdings besteht das grundsätzliche Problem weiterhin. Im Dezember 1993 veröffentlichte die IETF (Internet Engineering Task Force) eine Ausschreibung für Angebote für ein neues Internet-Protokoll (RFC 1550). Einer der Vorschläge, die daraufhin eingingen, wurde 1994 als SIPP (Simple Internet Protocol Plus) in RFC 1710 veröffentlicht. 1995 wurde dieses Protokoll in IPv6 umbenannt und mit einigen Änderungen in RFC 1883 veröffentlicht. Der aktuelle Stand von 1998 ist in RFC 2460 beschrieben.

Anmerkung: Auch die Versionsnummern 5,7,8 und 9 wurden vergeben. Die Nummer 5 wurde dem experimentellen Internet-Stream-Protocol 2 (IS-II) (RFC 1819) zugeteilt, das einen garantierten Dienst auf der Vermittlungsschicht bereitstellen sollte. Nummer 7 wurde an TP/IX (RFC1475) vergeben, Nummer 8 an PIP (RFC 1621) und Nummer 9 an TUBA (RFC 1347). Keins dieser Protokolle wird jedoch heute verwendet.

2.2 Designentscheidungen

Die wichtigste Änderung gegenüber seinem Vorgänger hat IPv6 in der Vervierfachung seiner Adresslänge erfahren, statt 32 Bit stehen jetzt 128 Bit zur Verfügung. Die Anzahl der theoretisch zur Verfügung stehenden Adressen wurde also von etwa 4 Milliarden auf über 340 Sextillionen erhöht. Damit

werden Notlösungen wie *Network Address Translation* (NAT), *Port Address Translation* (PAT) und *Classless Inter-Domain Routing* (CIDR) überflüssig.

Daneben sollte aber auch das Protokoll vereinfacht werden, damit Router die Pakete schneller verarbeiten können. Dazu tragen die folgenden Maßnahmen bei:

Eine Fragmentierung von IP-Paketen auf dem Pfad wird ausgeschlossen. Wenn ein Router ein Paket erhält, das größer ist als die *Maximum Transfer Unit* (MTU) auf dem nächsten Leitungsstück, soll er dieses verwerfen und eine ICMPv6-Nachricht (siehe Abschnitt 2.4) an den Sender des Paketes schicken. Darin soll er ihn anweisen, von nun an kleinere Pakete zu übermitteln. Die Aufgabe der Fragmentierung wird also an die Endknoten delegiert.

Es wird keine Prüfsumme des Paketes mehr berechnet oder übertragen. Die Fehlererkennung wird also den Protokollen auf höheren und niedrigeren Schichten überlassen. Sowohl Ethernet (IEE 802.3) als auch TCP (RFC 793) und UDP (RFC 768) berechnen eine Prüfsumme und übertragen sie.

Ebenfalls mit dem Ziel einer schnelleren Verarbeitungsgeschwindigkeit, wurde die Entscheidung getroffen, dass IPv6-Header eine feste Größe haben sollen. Die Größe des IPv4-Header war nämlich aufgrund des *Optionen*-Feldes variabel, was bei der Auswertung des Paketes zu recht aufwendigen Fallunterscheidungen führte. Das *Optionen*-Feld wird also gestrichen und durch den folgenden Mechanismus ersetzt:

Auf den eigentlichen IPv6-Header, der immer 40 Byte lang ist, (zum Vergleich, der IPv4 Header ist mindestens 20 Byte lang) können beliebig viele so genannte Erweiterungsheader folgen. Jeder Teilheader verweist im Feld „*Nächster Header*“ auf den Typ des jeweils nächsten. Ein Router kann so schnell entscheiden, welche Optionen ihn nichts angehen und diese ignorieren. Falls kein Header mehr folgt, benennt dieses Feld das Protokoll der Transportschicht, an das dieses IP-Paket übergeben wird (etwa TCP, UDP oder ICMPv6).

Neben der Vereinfachung sollten aber auch Diensttypen und Datenflüsse, die etwa für Echtzeitanwendungen wie Internettelefonie wichtig sind, besser unterstützt werden. Jedes Paket kann dafür eine Flussnummer erhalten. So können Router diejenigen Pakete, die zum selben Datenfluss gehören, im-

mer gleich behandeln. Damit soll erreicht werden, dass die Varianz der Latenz möglichst klein wird, da diese häufig besonders störend wirkt. Auf diesem Wege hat man versucht die Vorteile einer verbindungsorientierten mit denen einer paketbasierten Vermittlung zu vereinen.

Darüber hinaus waren auch Sicherheitsüberlegungen während des Designs des Protokolls von Bedeutung. IPSec wurde zunächst für IPv6 entwickelt und unterstützt sowohl die Verschlüsselung als auch die Authentifizierung des Datenverkehrs unterstützt. Inzwischen hat diese Technik allerdings auch Einzug in die IPv4-Welt gefunden und soll deshalb nicht Teil dieser Arbeit sein.

2.3 Der IPv6-Header

2.3.1 Der Hauptheader

Das Ergebnis dieser Überlegungen – der IPv6-Hauptheader – wird in der folgenden gezeigt:

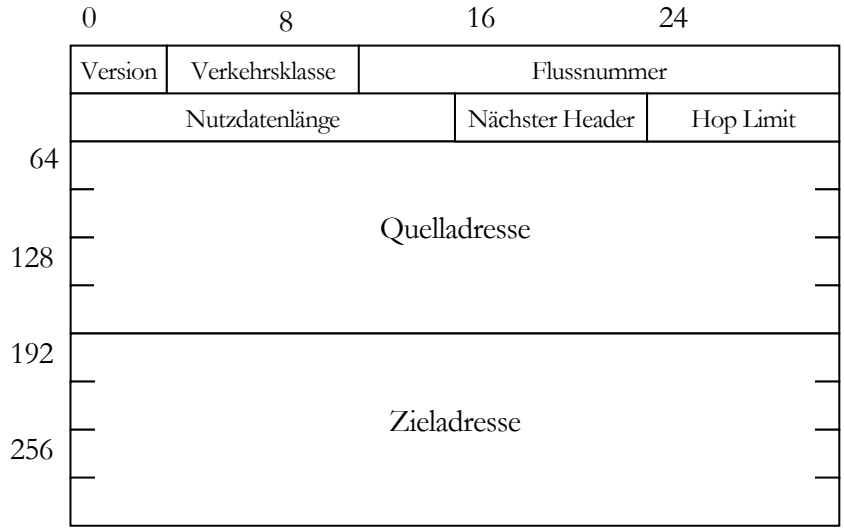

Abbildung 2.1 Der IPv6-Hauptheader

Die folgende Tabelle beschreibt die einzelnen Felder:

Feld	Länge	Bedeutung
Version	4 Bit	IP-Versionsnummer (= 6)

Verkehrsklasse	8 Bit	Für Quality of Service (QoS) verwendeter Wert. Eine Art Prioritätsvergabe.
Flussnummer	20 Bit	Ebenfalls für QoS oder Echtzeitanwendungen verwendeter Wert. Pakete die eine Flussnummer tragen werden alle gleich behandelt.
Nutzdatenlänge	16 Bit	Länge des IPv6-Paketinhaltes in Byte (ohne Header aber inklusive der Erweiterungsheader)
Nächster Header	8 Bit	Identifiziert den Typ des nächsten Erweiterungs-Headers
Hop-Limit	8 Bit	Maximale Anzahl an Zwischenschritten über Router, die ein Paket zurücklegen darf; wird beim Durchlaufen eines Routers („Hops“) um Eins verringert. Pakete mit Null als Hop Limit werden verworfen
Quelladresse	128 Bit	Adresse des Senders
Zieladresse	128 Bit	Adresse des Empfängers

2.3.2 Die Erweiterungsheader

IPv6-Nutzdaten können Null oder mehrere Erweiterungsheader unterschiedlicher Länge enthalten. Das Feld **Nächster Header** in einem IPv6-Header gibt den nächsten Erweiterungsheader an. Jeder Erweiterungsheader enthält wiederum das Feld **Nächster Header**, das den nächsten Erweiterungsheader angibt. Der letzte Erweiterungsheader gibt das Protokoll einer höheren Schicht an (etwa TCP, UDP oder ICMPv6), an das der Inhalt des Paketes übergeben werden soll.

Der IPv6-Header und die Erweiterungsheader ersetzen den IPv4-Header und seine Fähigkeit, Optionen anzugeben. Das neue Format für Erweiterungsheader erlaubt es, IPv6 um die Unterstützung künftiger Anforderungen und Fähigkeiten zu ergänzen. Im Unterschied zu den Optionen im IPv4-Header kennen IPv6-Erweiterungsheader keine Größenbeschränkungen und können alle Erweiterungsdaten aufnehmen, die für die IPv6-Kommunikation benötigt werden.

RFC 2460 definiert die folgenden IPv6-Erweiterungsheader, die alle IPv6-Knoten unterstützen müssen:

Bezeichnung	Beschreibung
Optionen für Teilstrecken	Verschiedene Informationen, die von jedem Router auf dem Weg verarbeitet werden. Derzeit gibt es nur die Option für Jumbogramme, also Pakete mit einer Größe von mehr als 65.536 Byte.
Optionen für Ziele	Zusätzliche Informationen für das Ziel. Derzeit sind keine solche Option definiert.
Routing	Führt bestimmte Router auf, die Pakete auf ihrem Weg zum Ziel durchlaufen müssen, zusätzliche Router sind aber erlaubt. Entspricht also etwa dem Loose Source Routing von IPv4.
Fragmentierung	Behandelt Fragmentierung in einer ähnlichen Weise wie IPv4. Er enthält den Datagramm-Bezeichner, die Fragmentnummer und ein Bit, das bezeichnet, ob weitere Fragmente folgen. Wie bereits in Abschnitt 2.2 erwähnt, darf nur ein Endknoten die Fragmentierung vornehmen.
Authentifizierung	Bietet einen Mechanismus, durch den der Empfänger eines Paketes sicher sein kann, wer es gesendet hat.
Verschlüsselte Sicherheitsdaten	Ermöglicht die Verschlüsselung eines Paketes, so dass nur der beabsichtigte Empfänger es lesen kann.

Typische IPv6-Pakete enthalten keine Erweiterungsheader. Sendende Hosts fügen einen oder mehrere Erweiterungsheader nur dann ein, wenn zwischengeschaltete Router oder der Zielknoten das Paket auf spezielle Weise behandeln müssen.

2.4 ICMPv6

Bei IPv4 wurden Informations- und Fehlernachrichten mithilfe des *Internet Control Message Protokolls* (ICMP, RFC 792) übermittelt. Diese Aufgabe und noch einige andere übernimmt bei IPv6 das Protokoll ICMPv6.

Der Aufbau einer ICMPv6 Paketes entspricht dabei genau dem einer ICMP-Nachricht.

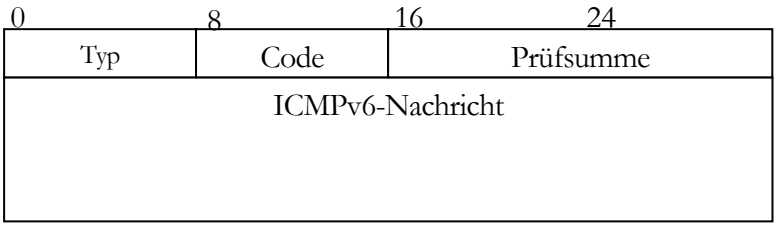

Abbildung 2.2 Aufbau eines ICMPv6-Paketes

Neben den üblichen Aufgaben, die auch ICMP schon übernommen hatte (etwa Fehlerbenachrichtigung bei Nicht-Erreichbarkeit eines Hosts, Echo-Requests und ICMP-Redirects), wird ICMPv6 auch für die *Nachbarschaftserkennung* (bei IPv4 wird dazu ARP verwendet, RFC 2461), die *Gruppenverwaltung* (statt IGMP bei IPv4, wird in Abschnitt 2.6.2.2 behandelt) und *Routerankündigungen* (siehe Abschnitt 3.3) verwendet.

2.5 Adress-Notation

Eine IPv6-Adresse ist 128 Bit lang. Dies führte dazu, dass man sich ein Format zur Darstellung überlegen musste, das einigermaßen menschenlesbar bleibt. Wäre man bei der Dotted-Decimal-Darstellung, wie sie bei IPv4 üblich ist, geblieben, sähe eine übliche IPv6 wie folgt aus:

Beispiel: 35.238.0.18.0.0.0.0.0.0.163.0.0.174.255

Stattdessen unterteilt man die Adresse in 8 Blöcke zu je 2 Byte, die in hexadezimaler Schreibweise und getrennt von Doppelpunkten notiert werden. Die obige Adresse notiert man also als:

Beispiel: 23EE:0012:0000:0000:0000:00A3:0000:AEFF

Außerdem sind die Vereinfachungen möglich:

Führende Nullen in jedem Block können weggelassen werden. Es muss aber mindestens eine Null stehen bleiben.

Beispiel: 23EE:12:0:0:0:A3:0:AEFF

Eine Folge von Blöcken, die nur aus Nullen besteht, kann durch :: (doppelter Doppelpunkt) ersetzt werden. Mehrere Ersetzungen sind nicht möglich, da sie zu Mehrdeutigkeiten führen würden.

Beispiel: 23EE:12::A3:0:AEFF

2.6 Adresstypen

IPv6 verfügt über die folgenden drei Typen von Adressen:

- *Unicast-Adressen* bezeichnen eine einzelne Schnittstelle.
- *Multicast-Adressen* bezeichnen eine Gruppe von Schnittstellen
- *Anycast-Adressen* bezeichnen die nächste Schnittstelle innerhalb einer Gruppe

Die einzelnen Adresstypen werden im Folgenden beschrieben.

2.6.1 Unicast-Adressen

Eine Unicastadresse kennzeichnet eine einzelne Schnittstelle. Eine solche Adresse wird für die Kommunikation von einer Quelle mit einem einzelnen Ziel (1:1) verwendet.

IPv6-Unicastadressen unterteilen sich in die folgenden 4 verschiedenen Typen:

- *Globale Unicastadressen* , die global eindeutig sind geroutet werden können,
- *Linklokale Adressen*, die nur lokal in einem Subnetz eindeutig sind und nie geroutet werden,
- *„Unique Local"-Adressen,* die die Rolle der privaten Adressen übernehmen und zwar geroutet werden dürfen, aber nicht global eindeutig sind, und

- *Spezielle IPv6-Adressen* , die die unspezifizierte und die Loobback-Adresse beinhalten.

2.6.1.1 Globale Unicastadressen

Globale Unicastadressen sind mit öffentlichen IPv4-Adressen vergleichbar d.h. sie sind global routebar. Außerdem können sie aggregiert werden, um eine effiziente Routinginfrastruktur zu erzeugen. Das IPv4-Internet ist eine Mischung aus hierarchischem und flachem Routing, das auf IPv6 basierende Internet hingegen wurde von Grund auf dazu entworfen, eine effiziente und hierarchische Adressierung sowie ein effizientes und hierarchisches Routing zu unterstützen. Globale Unicastadressen sind darüber hinaus global eindeutig. In der folgenden Abbildung wird die allgemeine Struktur einer globalen Unicastadresse veranschaulicht, wie sie in RFC 3587 definiert ist.

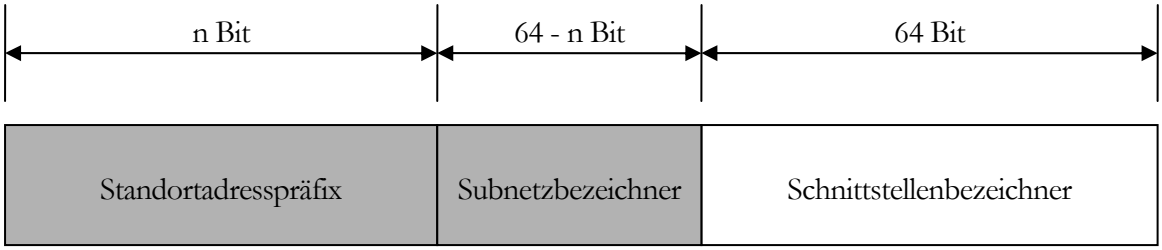

Abbildung 2.3 Allgemeiner Aufbau von globalen Unicastadressen

In der folgenden Abbildung wird die Struktur jener globalen Unicastadressen veranschaulicht, die von der *Internet Assigned Numbers Authority* (IANA) zum Zeitpunkt der Erstellung dieser Arbeit zugewiesen werden.

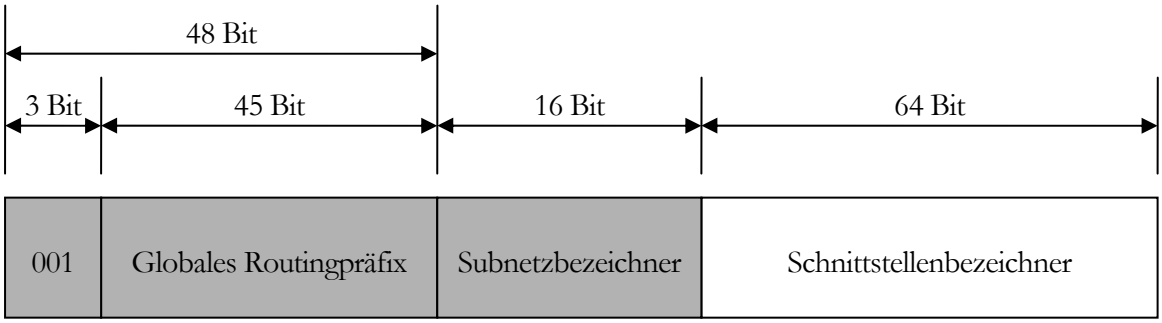

Abbildung 2.4 Aufbau von globalen Unicastadressen, wie sie im Moment vergeben werden

Folgende Felder werden in globalen Unicastadressen verwendet:

Adress-Feld	Länge	Beschreibung
Fester Anteil	3 Bit	Der feste Anteil ist auf 001 festgelegt. Das Adresspräfix für momentan zugewiesene globale Adressen lautet also 2000::/3.
Globales Routingpräfix	45 Bit	Das globale Routingpräfix kennzeichnet den Standort einer bestimmten Organisation. Die Kombination aus den drei festen Bit und dem aus 45-Bit bestehenden globalen Routingpräfix wird zum Erstellen eines 48-Bit-Standortadresspräfixes verwendet, das den einzelnen Standorten einer Organisation zugewiesen wird. Nach der Zuordnung leiten die Router im IPv6-Internet den IPv6-Datenverkehr, der mit dem aus 48 Bit bestehenden Adresspräfix übereinstimmt, an die Router des Standorts der Organisation weiter.
Subnetz-Bezeichner	16 Bit	Der Subnetz-Bezeichner kennzeichnet die Subnetze am Standort einer Organisation. Die Organisation kann mithilfe dieser 16 Bit 65.536 Subnetze an einem Standort oder mehrere Ebenen einer Adressierungshierarchie sowie eine effiziente Routinginfrastruktur erstellen.
Schnittstellen-ID	64 Bit	Der Schnittstellen-Bezeichner kennzeichnet eine Schnittstelle im Subnetz eines Standtortes.

Beispiel: Die Adresse 2007:4711:AFFE:1337:FB4:D00F:FEE:112 ist z. B. eine globale IPv6-Unicastadresse. Innerhalb dieser Adresse gilt Folgendes:

Adress-Anteil	Beschreibung
2007:4711:AFFE	Standort der Organisation
1337	Subnetz an diesem Standort
FB4:D00F:FEE:112	Schnittstelle im Subnetz an diesem Standort

Die Felder in der globalen Unicastadresse erstellen also eine Struktur mit drei Ebenen.

2.6.1.2 Linklokale Adressen

Netzwerkknoten können auch ohne global-erreichbare Adressen mit anderen Knoten kommunizieren, die an das gleiche Subnetzwerk angeschlossen sind.

Dazu wird ihnen eine so genannte linklokale Adresse zugewiesen. Der Vorgang dieser Zuweisung wird in Abschnitt 3.1 behandelt. Linklokale Adressen sind nur auf dem aktuellen Netzwerk-Segment eindeutig und können nur dort verwendet werden um andere Knoten zu erreichen, daraus folgt auch, dass Router Nachrichten, die solche Adressen als Absender oder Empfänger enthalten niemals weiterleiten dürfen.

Eine solche Adresse ist für die Prozesse zur Nachbarerkennung (Neighbor Discovery) erforderlich und wird immer automatisch konfiguriert, auch wenn eine globale Unicastadresse nicht benötigt wird.

In Abbildung 2.5 wird die Struktur einer linklokalen Adresse veranschaulicht.

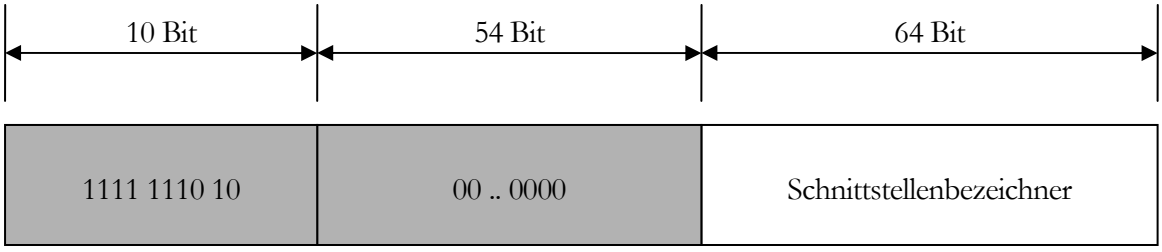

Abbildung 2.5 Aufbau einer linklokalen Unicast-Adresse

Da die ersten 64 Bit der verbindungslokalen Adresse feststehend sind, wird als Adresspräfix für alle verbindungslokalen Adressen FE80::/64 verwendet.

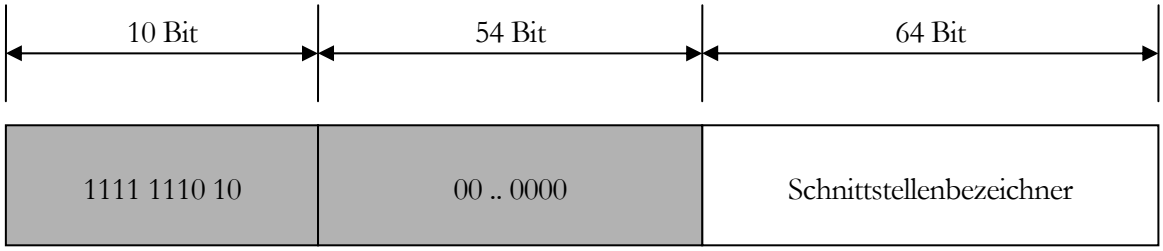

Abbildung 2.6 Aufbau von linklokalen Adressen

Anmerkung: Seit 1998 unterstützen sowohl Apple- als auch Windowsrechner linklokale **IPv4**-Adressen. Dafür wurde der Adressbereich 169.254.0.0/16 von der IANA reserviert. Adressen aus diesem Bereich sollen weder manuell noch von DHCP-Servern vergeben werden. Bei Apple heißt das zugehörige Protokoll Bonjour (früher Rendevous) bei Windows APIPA (*Automatic Private IP Addressing*). Mittlerweile wird das Verfahren in RFC 3927 beschrieben und dort als *Dynamic Configuration of IPv4 Link-Local Addresses* bezeichnet. Dabei wird das 16-Bit-lange Postfix mittels eines Pseudozufalls-Generators erstellt. Dann wird mithilfe von ARP geprüft, die daraus resultierende Adresse auf dem Link verwendet wird. Falls nicht wird die linklokale Adresse angenommen, ansonsten wird der Vorgang wiederholt.

2.6.1.3 „Unique Local"-Adressen

RFC 4193 spezifiziert sogenannte „Unique Local"-Adressen, die die Rolle der drei privaten Adressbereiche von IPv4 (10.0.0.0/8, 172.16.0.0/12 und 192.168.0.0/16) übernehmen sollen. Sie sind also für den lokalen Einsatz beispielsweise innerhalb eines Firmen- oder Heimnetzes entworfen, haben aber zusätzlich noch einigen entscheidenden Vorteil: Sie sind so entworfen, dass sie selbst wenn sie versehendlich nach Außen geroutet werden sollten oder wenn zwei oder mehrere solcher Netze per Tunnel miteinander verbunden werden, höchst wahrscheinlich kein Konflikt auftritt. Dies wird erreicht, indem an ein festes Präfix (das eine einfache Filterung an Border-Routern erlaubt) eine 40 Bit lange Zufallszahl gehängt wird. Die folgende Abbildung illustriert den Aufbau dieser Adressen:

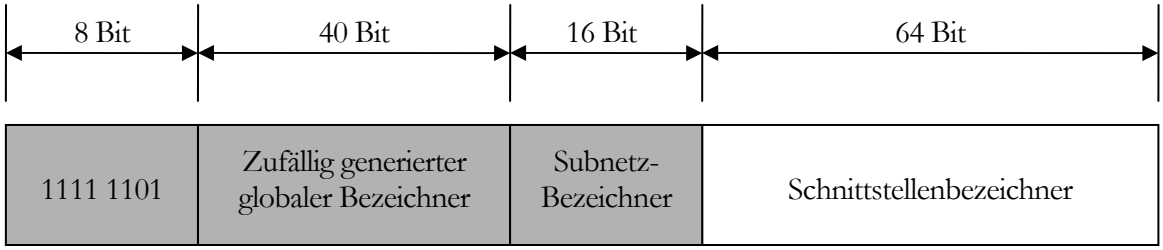

Abbildung 2.7 Aufbau von „Unique Local"-Adressen

2.6.1.4 Spezielle IPv6-Adressen

Darüber hinaus sind noch die folgenden beiden spezielle Adressen spezifiziert:

- Die *unspezifizierte Adresse* (0:0:0:0:0:0:0:0 oder ::) zeigt an, dass keine Adresse vorhanden ist und sie entspricht der nicht spezifizierten Adresse 0.0.0.0 von IPv4. Die nicht spezifizierte Adresse wird normalerweise als Quelladresse für Pakete verwendet, die versuchen, die Eindeutigkeit einer vorläufigen Adresse zu bestätigen. Die nicht spezifizierte Adresse wird niemals einer Schnittstelle zugewiesen oder als Zieladresse verwendet.

- Die *Loopbackadresse* (0:0:0:0:0:0:0:1 oder ::1) kennzeichnet eine Loopbackschnittstelle. Mithilfe dieser Adresse kann ein Knoten Pakete an sich selbst senden und sie entspricht der Loopbackadresse 127.0.0.1 von IPv4. An die Loopbackadresse gerichtete Pakete werden niemals über eine Verbindung gesendet oder von einem IPv6-Router weitergeleitet.

2.6.2 Multicast

Neben normalen Unicast-Adressen bietet IPv6 wie schon IPv4 die Möglichkeit mithilfe einer einzelnen Multicast-Adresse eine ganze Gruppe von Knoten zu erreichen.

Multicasting funktioniert ähnlich wie das Abonnieren einer Zeitschrift. Eine solche wird nur den Abonnenten zugestellt und genauso wird der Datenverkehr, der an eine für eine Multicastgruppe reservierte Adresse geschickt wurde, nur von den Netzwerkknoten empfangen und verarbeitet, die Mitglied eben dieser Gruppe sind. Als Multicastgruppe wird eine Gruppe von Hosts bezeichnet, die gemeinsam eine bestimmte Multicastadresse verwenden.

Beim Multicasting gibt es noch andere wichtige Aspekte:

- Die Gruppenmitgliedschaft ist dynamisch. Hosts können also einer Gruppe zu einem beliebigen Zeitpunkt beitreten oder sie wieder verlassen.
- Der Beitritt zu einer Multicastgruppe erfolgt über das Senden von Gruppenmitgliedschaftsnachrichten. In IPv6 werden Multicast Listener Discovery-Nachrichten (MLD, siehe Abschnitt 2.6.2.2) verwendet, um die Gruppenmitgliedschaft auf einem Link zu bestimmen.
- Die zulässige Größe der Gruppen unterliegt keiner Beschränkung, und die Mitglieder können über mehrere Netzwerkabschnitte verteilt sein (wenn die Übermittlung von Multicastdatenverkehr und Informationen zur Gruppenzugehörigkeit durch Routerverbindungen unterstützt wird).
- Datenverkehr kann von einem Host an die Adresse der Gruppe gesendet werden, ohne dass es sich beim Host um ein Mitglied der betreffenden Gruppe handeln muss.

2.6.2.1 Multicast-Adressen

Die ersten 16 Bit einer Multicast-Adresse lauten immer FF, darauf folgen ein mit „Flags“ bezeichnetes Feld und ein Feld, das „Scope“ genannt wird. Darauf folgt ein Feld, das die Gruppe bezeichnet. Die folgende Abbildung verdeutlicht diesen Aufbau.

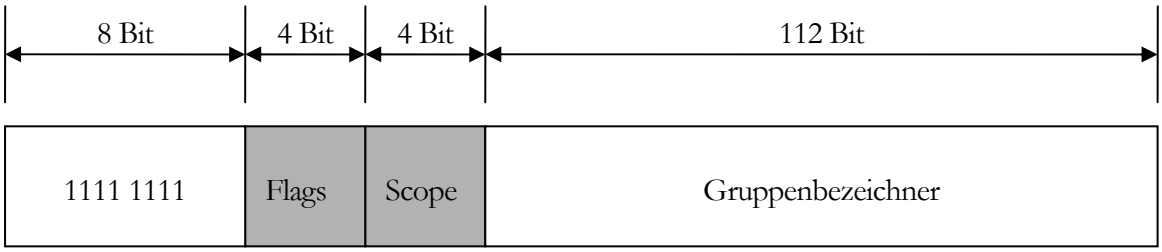

Abbildung 2.8 Aufbau einer IPv6-Multicast-Adresse

Feld	Beschreibung
Flags	Kann die Werte 0 und 1 annehmen. 0 steht dabei für eine permanente Ad-

	resse (well-know) und 1 für temporäre (transient). Die anderen Werte sind reserviert.
Scope	Steht für die Reichweite der Multicast-Adresse, d.h. wie weit sie weitergeleitet werden. Der Wert 1 steht für interface-local (verlässt nie die Schnittstelel), 2 für linklokal (wird nie geroutet). Die Werte 4 bis D stehen im Grunde Administratoren zur Verfügung um eigene Multicast-Reichweiten zu erstellen (4 heißt *admin-local*, 5 *site-local* (sollen nicht von Border-Routern geroutet werden) und 8 *organisation-local*). E steht für globale Reichweite. 3 und F sind für die zukünftige Nutzung reserviert.
Gruppenbezeichner	Gibt die Multicast-Gruppe an

IPv6 kennt keinen Broadcast, diese Funktionalität wird ebenfalls von Multicast bereitgestellt. Die IANA definiert dafür folgende Multicast-Adressen[1]:

- FF02::1 bezeichnet alle Knoten am aktuellen Link
- FF02::2 bezeichnet alle Router am aktuellen Link

Darüberhinaus sind dort noch andere Adressen wie etwa FF02::1:2 für alle DHCP-Server am Link definiert.

Bei der Übertragung via Ethernet wird die Multicast-Adresse auf eine MAC-Adresse abgebildet. Dazu werden die letzten 4 Byte an das Präfix 33-33 angehängt, so dass eine 6-Byte-lange MAC-Adresse entsteht. So kann schon die Netzwerkkarte relevante von unrelevanten Paketen unterscheiden. Die folgende Abbildung verdeutlicht diese Zuordnung

[1] http://www.iana.org/assignments/ipv6-multicast-addresses

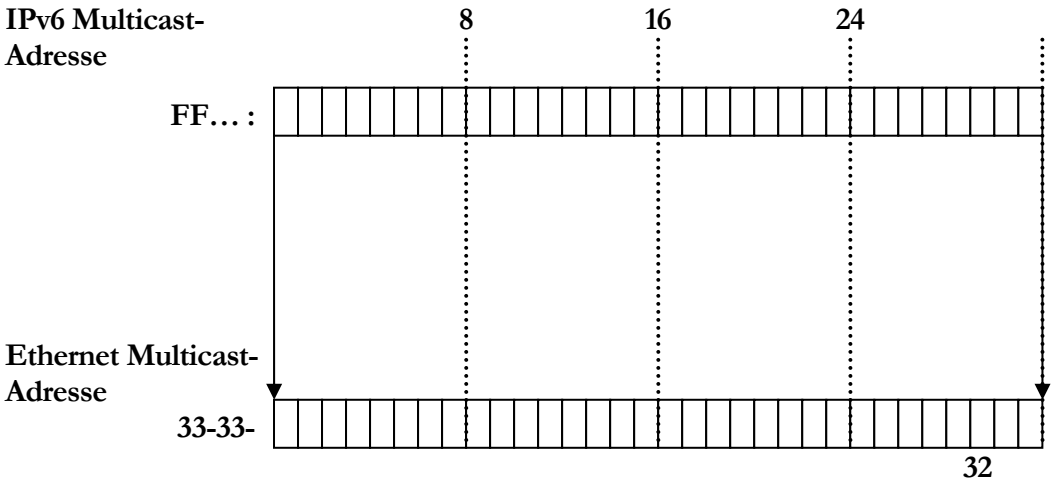

Abbildung 2.9 Abbildung von IPv6 Multicast-Adressen auf Ethernet Multicast-Adressen

Diese Abbildung ist aber selbstverständlich nicht injektiv, so dass noch eine weitere Filterung stattfinden muss.

Beispiel: Die Multicastadresse FF02::1 für alle Knoten im linklokalen Bereich wird der Ethernet-Multicastadresse 33-33-00-00-00-01 zugeordnet.

Beispiel: Die Beispieladresse FF02::1:FF3F:2A1C des angeforderten Knotens wird der Ethernet-Multicastadresse 33-33-FF-3F-2A-1C zugeordnet.

2.6.2.2 MLD-Nachrichten

Multicast Listener Discovery (MLD) ist die Entsprechung für IGMP aus der IPv4-Welt – baut aber, statt ein eigenes Protokoll zu definieren auf ICMPv6 auf. RFC 2710 spezifiziert es zum Austausch von Informationen über den Status der Gruppenmitgliedschaften zwischen Routern und den einzelnen Knoten. Ob noch Knoten Mitglied einer gegebene Gruppe sind, wird dabei von den einzelnen Netzwerkknoten ermittelt und der Status wird periodisch von den multicastfähigen Routern abgefragt. Für weitere Informationen sei auf den entsprechenden RFC verwiesen.

2.6.3 Anycast

Neben Unicast und Multicast bietet IPv6 noch einen weiteren (und im Vergleich zu IPv4 neuen) Adresstyp an. Eine Anycastadresse kennzeichnet (wie Multicastadressen) mehrere Schnittstellen, allerdings

wird eine Nachricht mit einer solchen Zielangabe (wie Unicast-Nachrichten) nur an eine Schnittstelle ausgeliefert, wobei es sich um die nächstliegende handelt. Als "nächstliegende" Schnittstelle wird die Schnittstelle mit der geringsten Routingentfernung (dem kürzesten Pfad) bezeichnet. Eine Anycastadresse wird für die Kommunikation zwischen einer Quelle und einem Ziel aus einer Gruppe von mehreren Zielen verwendet.

2.6.3.1 Anycast-Adressen

Anycast-Adressen sind formal Unicast-Adressen, dürfen aber nur als Ziel und nicht als Quelladresse verwendet werden.

Subnetzrouter-Anycastadressen sind spezielle Anycast-Adressen, die allen Routerschnittstellen, die mit einem gegebenen Subnet verbunden sind, zugeordnet werden.

Die ersten 64 Bit einer solchen Adresse entsprechen dem Subnetzpräfix, die zweiten 64 Bit erhalten den Wert 0. Die Subnetzrouter-Anycastadresse kann für die Kommunikation mit einem von mehreren Routern verwendet werden, die mit einem Remotesubnetz verbunden sind, z. B. zum Abrufen einer Statistik zur Netzwerkverwaltung für den Datenverkehr im Subnetz.

3 Zustandslose Autokonfiguration

Ohne zugewiesene Adressen ist eine Kommunikation innerhalb eines IP-Netzwerkes unmöglich. Bei IPv4 gab es nur zwei Möglichkeiten an verwendbare Adressen zu: kommen, die manuelle Konfiguration und die Vergabe mittels DHCP. Dieses Verfahren wird als zustandsbehaftet (stateful) bezeichnet, weil der DHCP-Server über die vergebenen Adressen Buch führt. In vielen Fällen ist dieses Verfahren allerdings überdimensioniert und könnte durch ein einfacheres Verfahren ersetzt werden. IPv6 bietet eine solche Möglichkeit an, die zustandslose Autokonfiguration.

Dabei werden zunächst linklokale Adressen von jedem Netzwerkknoten fast vollständig autonom generiert. Dies ist insbesondere aufgrund des großen verfügbaren Adressraumes einfach möglich. Danach kann von einem Router ein global gültiges Präfix übermittelt werden. Daraus wird dann unabhängig vom Router und, ohne dass dieser davon weiß oder darüber Buch führen müsste, eine global eindeutige Adresse zusammengesetzt. Das es hierbei nicht zu Kollisionen kommt, ist wiederum dem großen verfügbaren Adressraum zu verdanken. Dabei kann die eigentlich ja global schon eindeutige MAC-Adresse in die IPv6-Adresse eingebettet werden, daraus ergeben sich fast immer kollisionsfreie Adressvergaben. Die einzelnen Schritte einer solchen Konfiguration werden nun im Einzelnen erläutert. Danach wird noch auf die Vorgehensweise eingegangen, wenn die so konfigurierten Adressen ungültig sind und welche Maßnahmen getroffen werden können um eine Rückverfolgung auf einen einzelnen Computer anhand einer automatisch generierten Adresse möglichst zu vermeiden.

3.1 Generierung von linklokalen Adressen

Ein Knoten bildet eine linklokale Adresse für eine Schnittstelle, sobald diese aktiviert wird. Dabei ist es egal, ob man die Schnittstelle später automatisch oder manuell konfigurieren möchte. Die linklokale Adresse wird immer und für jede Schnittstelle unabhängig gebildet. Sie ist erforderlich, um eine eventuelle anschließende zustandslose oder zustandsbehaftete automatische Konfiguration überhaupt zu ermöglichen. Außerdem kommunizieren Rechner innerhalb eines Subnetzes oftmals über die linklokale Adresse anstatt über routebare, globale Adressen, wenn von den oberen Schichten nicht explizit eine

zu verwendende Adresse vorgegeben wird. Die linklokale Adresse wird aus dem 64 Bit langen EUI64 Schnittstellen-Bezeichner und dem 64-Bit langen Präfix für linklokale Adressen FE80::0/64 gebildet. Eine linklokale Adresse hat also die folgende Form:

FE80:0000:0000:0000:EUI64/64

Die EUI64 Adresse wird im Falle von Ethernet als Sicherungsschicht aus der MAC-Adresse der Schnittstelle gebildet. Dieses Verfahren wird in RFC 2373 beschrieben. Dabei werden in die Mitte der 48 Bit langen MAC-Adresse die zwei Bytes FF:FE eingeschoben, um auf 64 Bit zu kommen. Außerdem wird das siebte Bit der MAC-Adresse, welches auch das Universal/Local Bit genannt wird, auf den Wert 1 gesetzt.

Abbildung 3.1 zeigt den Aufbau einer MAC-Adresse:

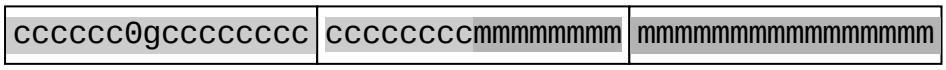

Abbildung 3.1 Aufbau einer MAC-Adresse

```
c: company id
0: universal/local Bit (0 = global scope, 1 = local)
g: individual/group Bit (0 = unicast, 1 = Gruppe)
m: manufacturer-selected extension identifier
```

Eine daraus generierte EUI64-Adresse:

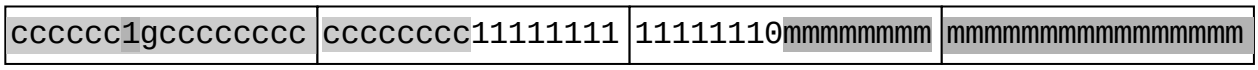

Abbildung 3.2 Aufbau einer EUI-64 Adresse aus einer MAC-Adresse

Aus der MAC-Adresse 00:11:09:8D:23:1F wird die linklokale Adresse fe80::211:9ff:fe8d:231f/64.

Bevor diese linklokale Adresse jetzt der Schnittstelle endgültig zugewiesen werden kann, muss zunächst geprüft werden, ob sie im Subnetz noch nicht verwendet wird. Dies erfolgt mit der *Duplicate Address Detection* (DAD).

3.2 Erkennung duplizierter Adressen (DAD)

Mithilfe der *Duplicate Address Detection* (DAD) wird überprüft, ob eine Adresse bereits verwendet wird. Dazu wird ICMPv6 verwendet, das auch die Funktionalität von ARP aus der IPv4 Welt liefert. Die DAD muss für alle Unicast-Adressen einzeln durchgeführt werden, egal ob sie manuell oder mittels einer automatischen Methode festgelegt wurden. Falls dabei festgestellt wird, dass die gewünschte Adresse schon verwendet wird, kann diese der Schnittstelle nicht zugewiesen werden und in der Regel muss nun eine manuelle Konfiguration erfolgen. Salopp gesprochen fragt der Knoten: „Wer hat die Adresse XY?“, und denkt sich dabei: „Hoffentlich antwortet niemand.“

Für den Zeitraum, in dem die DAD durchgeführt wird ist die generierte linklokale Adresse nur „tentative“ (schwebend, provisorisch) gültig. In diesem Zustand, verarbeitet die Schnittstelle nur Nachbarschafts-Anforderungen und -Ankündigungen unter dieser Adresse, da diese für dieses Verfahren benötigt werden. Alle anderen empfangen Pakete mit dieser Zieladresse werden verworfen. Dies ist deshalb wichtig, da es ja bereits einen Knoten geben kann, welcher dieselbe Adresse benutzt, und über sie kommuniziert.

Der Knoten könnte für die Kommunikation einfach die „Alle-Knoten“-Multicast-Adresse verwenden, dies würde dem unter IPv4 für ARP üblichen Broadcast-Kanal entsprechen, allerdings würde dies zu unnötig viel Last auf den anderen Knoten des Netzwerkes führen. Deshalb bildet er zu Beginn der DAD bildet ein Knoten eine so genannte *Solicited Node Multicast*-Adresse und tritt der entsprechenden Gruppe bei. Eine wichtige Eigenschaft einer solchen Adresse ist, dass sie eindeutig von der zu testenden Adresse abhängt. Möchte also später ein anderer Knoten dieselbe Adresse verwenden, wird er auch die gleiche Multicast-Adresse generieren. Außerdem ist es wichtig, dass der Knoten die Gruppe auch nach erfolgreicher Zuweisung der nicht verlässt, damit es später seine Adresse gegenüber einem anderen Knoten „verteidigen“ könnte. RFC 2373 beschreibt die Generierung einer solchen Adresse wie folgt: An das Präfix FF02::1:FF00:0/104 werden die letzten 24 Bit der zu testenden Adresse angehängt.

Beispiel: Aus der linklokalen Adresse fe80::211:9ff:fe8d:231f wird die Multicast-Adresse ff02::1:FF8d:231f.

Nun sendet der Knoten eine so genannte Nachbarschaftsanfrage (*neighbor solicitation*, „Wer hat diese Adresse“) an diese Solicited-Node-Multicast-Adresse. In das Nachrichtenfeld des ICMPv6-Paketes trägt er die zu prüfende Adresse ein und als Quelle wird die unspezifizierte Adresse :: verwendet.

Falls ein Knoten mit einer Nachbarschaftsankündigung (*neighbor advertisement,* „XY hat diese Adresse“) antwortet und die Adresse bereits besitzt (steht dann im Quelladress-Feld des Datagramms), ist die DAD fehlgeschlagen und die entsprechende Adresse darf nicht verwendet werden.

Falls hingegen kein anderer Knoten die generierte Adresse besitzt, ist sie nicht mehr schwebend, sondern nunmehr gültig.

Im Falle einer linklokalen Adresse bekommt sie eine gültige und bevorzugte Lebenszeit von unendlich. Nun besitzt die Schnittstelle eine linklokale Adresse, mit der es innerhalb eines Subnetzes kommunizieren kann. Der Knoten damit in der Lage eine zustandlose Autokonfiguration für die Schnittstelle durchzuführen.

3.3 Generierung von globalen Adressen

Bei der zustandslosen Autokonfiguration von IPv6 Adressen werden globale Adressen gebildet, indem vor den Schnittstellenbezeichner ein global routebares Präfix gesetzt wird. Ein solches Präfix kann ein Knoten aus einer Routerankündigung – einer speziellen ICMPv6-Nachricht – erhalten.

Router-Ankündigungen werden periodisch von Routern an die „Alle Knoten“-Multicast-Adresse (FF02::1) gesendet. Ein Knoten kann eine Router-Ankündigung aber auch mit Hilfe einer *Router-Anfrage* (einer anderen ICMPv6-Nachricht) anfordern. Als Zieladresse trägt er dabei die „Alle-Router“-Multicast-Adresse (FF02::2) ein. Falls ein Subnetz keinen Router enthält, der diesen Dienst anbietet, muss der Knoten manuell mit einer globalen Adresse konfiguriert werden.

Die folgende Abbildung zeigt den Aufbau einer Router-Ankündigungs-Nachricht, wie er in RFC 2461 spezifiziert wird:

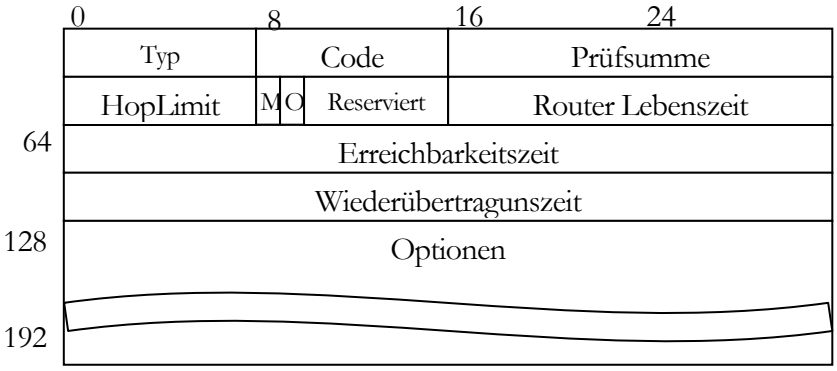

Abbildung 3.3 Aufbau einer Router-Ankündigungs-Nachricht

Die nachfolgende Tabelle beschreibt die einzelnen Felder:

ICMPv6-Feld	Beschreibung
Typ	= 134 (Router-Ankündigung)
Code	= 0
Hop Limit	= 255
M	„verwaltete Adresskonfiguration“-Schalter (siehe Abschnitt 4.2)
O	„andere zustandsbehaftete Konfiguration“-Schalter (siehe Abschnitt 4.2)
Router-Lebenszeit	Gibt die Lebenszeit des Routers in Sekunden an -- der Wert FFFF ergibt 18.2 h (der Wert 0 gibt an, dass es sich nicht um einen Default Router handelt)
Ereichbarkeitszeit	Zeit in ms, während der ein Knoten glaubt, dass er einen Nachbarn erreichen kann, nachdem er eine Erreichbarekeisbestätigung empfangen hat (ein Wert von 0 besagt: unspezifiziert)
Wiederübertragungszeit	Zeit in ms zwischen dem Senden von Nachbarschafts-Anfrage-

<table>
<tr><td></td><td colspan="2">Nachrichten (ein Wert von 0 besagt: unspezifiziert)</td></tr>
<tr><td>Optionen</td><td colspan="2">Dieses Feld variabler Länge kann eine oder mehrere der folgenden Optionen enthalten. Dabei können auch mehrere Präfix-Informationen übertragen werden</td></tr>
<tr><td></td><td>Option</td><td>Beschreibung</td></tr>
<tr><td></td><td>Quell-Sicherungsschicht-Adresse</td><td>Schicht-3-Adresse der Schnittstelle, welche die Router-Ankündigungsnachricht gesendet hat</td></tr>
<tr><td></td><td>MTU</td><td>Maximale Paketgröße, die über diesen Router versendet werden dürfen</td></tr>
<tr><td></td><td>Präfix-Information</td><td>Gibt das Präfix an, welches für die Autokonfiguration verwendet werden soll
Hier werden auch die Bevorzugte und Gültige Lebenszeit angegeben</td></tr>
</table>

Im Falle der zustandslosen Autokonfiguration müssen die beiden Schalter M und O den Wert 0 erhalten. Außerdem muss mindestens eine Präfix-Informations-Option in der Router-Ankündigungsnachricht enthalten sein.

Die folgende Abbildung zeigt den Aufbau einer solchen Präfix-Informations-Option:

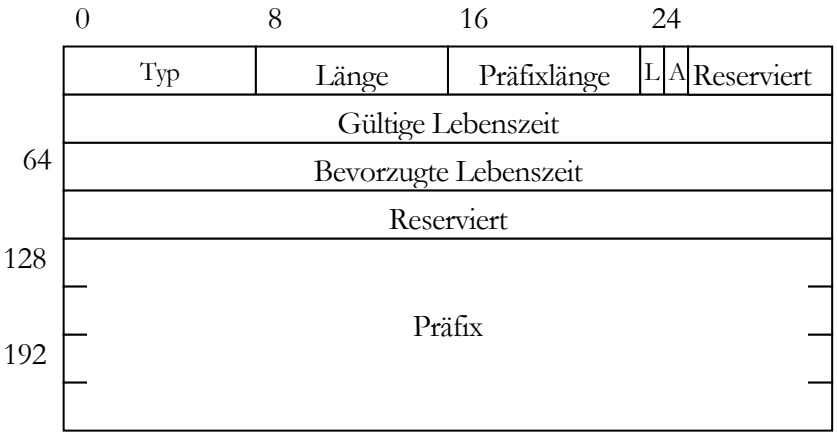

Abbildung 3.4 Aufbau der Präfix Information Option

Die folgende Tabelle beschreibt die einzelnen Felder:

Feld	Länge	Beschreibung
Typ	8 Bit	3 (Präfix Information Option)
Länge	8 Bit	4
Präfixlänge	8 Bit	Länge des Präfixes (üblicherweise 64)
L	1 Bit	„on-link“-Schalter - Wert 1: alle Knoten mit Adressen mit diesem Präfix sind direkt an dieses Subnetz angeschlossen - Wert 0: keine diesbezüglichen Informationen
A	1 Bit	„autonomous address-configuration“-Schalter -Wert 1: das Präfix kann für Autonome Adress Konfiguration genutzt werden (stateless) -Wert 0: die Präfix Information Option sollte ignoriert werden

Gültige Lebenszeit	32 Bit	Zeit in Sekunden, für die Adressen mit diesem Präfix gültig sind (Wert ffffffff: unendlich)
Bevorzugte Lebenszeit	32 Bit	Zeit in Sekunden, für die Adressen mit diesem Präfix bevorzugt sind (Wert ffffffff: unendlich)
Präfix	128 Bit	Präfix einer IP-Adresse (kein linklokales Präfix)

Die Gültige Lebenszeit muss mindestens so groß sein wie die Bevorzugte Lebenszeit, sonst wird die Option ignoriert.

Wenn der Knoten noch keine Adresse mit diesem Präfix in seiner Adressliste hat, das A-Bit in der Option gesetzt ist und die Gültige Lebenszeit nicht 0 ist, so bildet er eine neue Adresse mithilfe des Schnittstellen-Bezeichners (EUI-64).

Abbildung 3.5 verdeutlicht das Verfahren:

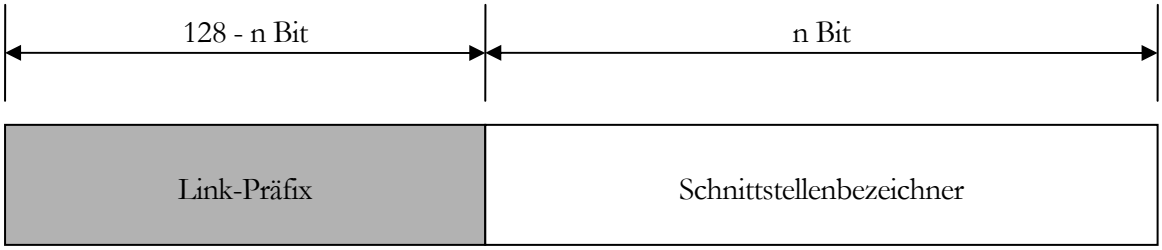

Abbildung 3.5 Aufbau einer automatisch gebildeten globalen Adresse

Wenn eine neue Adresse erfolgreich gebildet werden konnte (das heißt die erneut ausgeführte DAD erfolgreich war), wird sie in die Adressliste übernommen und erhält die in der Information-Präfix-Option angegebene Gültige und Bevorzugte Lebenszeit.

Falls es die gebildete Adresse bereits als automatisch konfigurierte Adresse gibt, werden nur die Lebenszeiten der Adresse aktualisiert.

3.4 Ablauf der Lebenszeit von Adressen

Sobald die Gültige Lebenszeit einer Adresse abgelaufen ist, wird aus der bevorzugten Adresse eine veraltete (deprecated) Adresse. Diese veraltete Adresse sollte ihre laufenden Verbindungen oder Übertragungen noch zu Ende bringen, aber keine neuen Kommunikationen mehr eingehen. Eingehende Pakete sollte noch bearbeitet werden.

Die Adresse wird ungültig, wenn die Gültige Lebenszeit abgelaufen ist. Eine ungültige Adresse darf von keiner Schnittstelle nicht mehr verwendet werden.

Die folgende Abbildung zeigt eine Übersicht über die Funktion der einzelnen Lebenszeit.

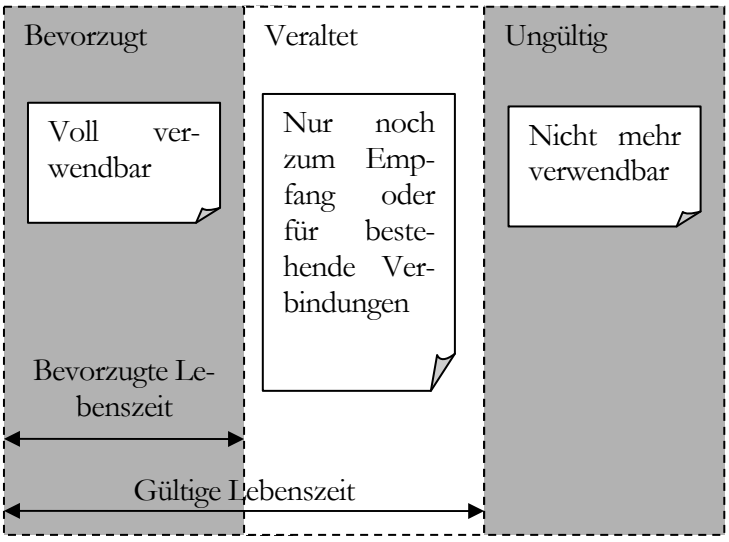

Abbildung 3.6 Überblick über die Lebenszeiten von Adressen

3.5 Generierung temporärer Adressen

Eine mithilfe des Verfahrens, das in diesem Kapitel vorgestellt wurde, generierte globale Adresse hat ein besonderes Kennzeichen: Die zweiten 64 Bit der Adresse Kennzeichnen in der Regel weltweit eindeutig eine bestimmte Schnittstelle und damit natürlich auch einen bestimmten Hostcomputer.

Dies kann aber in vielen Situationen besonders bei Privatanwendern zu unerwünschten Wirkungen führen. So könnte es zum einen möglich sein, die Bewegungen eines mobilen Gerätes zu verfolgen

und so ein Bewegungsprofil seines Besitzers zu erschließen. Außerdem könnten ansonsten unzusammenhängende Aktionen eines Benutzers auf verschiedenen Internetseiten korreliert werden und somit würden seine Privatsphäre und auch seine Anonymität im Netz praktisch aufgehoben. Um diesem Problem zu begegnen, spezifiziert RFC 3041 ein Verfahren für die Generierung von temporären Adressen. Dabei wird im Grunde genommen der Schnittstellenbezeichner zufällig generiert. Aufgrund des großen Adressraumes von 2^{64} sind Kollisionen fast ausgeschlossen. Sollte doch einmal eine Kollision auftreten, wird diese bei der obligatorischen DAD erkannt und dann werden neue Adressen generiert. Darüber hinaus besteht die Möglichkeit, die bevorzugte Lebenszeit einer solchen Adresse recht klein zu wählen, so dass die ausgehende Adresse für neue Kommunikationen sich häufig verändert und damit eine Rückverfolgung von einer Adresse auf einen Hostcomputer noch weiter erschwert wird.

4 DHCPv6

In diesem Kapitel wird DHCPv6 erläutert. Dabei wird zunächst auf dessen Einsatzmöglichkeiten eingegangen. Dann wird der Zusammenhang mit den im letzten Kapitel vorgestellten Router-Ankündigungsnachrichten hergestellt, worauf eine kurze Erklärung der Komponenten einer DHCPv6-Infrastruktur folgt. Danach wird erklärt, auf welchem Wege die einzelnen Komponenten miteinander kommunizieren und die beiden wichtigen Begriffe IA und DUID werden erläutert. Im darauf folgenden Abschnitt wird das Nachrichtenformat erläutert, das für die Kommunikation eingesetzt wird. Darauf folgt eine Beschreibung der verschiedenen Kommunikationsfälle. Abgeschlossen wird das Kapitel mit einer Erläuterung des Konzeptes der Präfixdelegation und von Relay-Agenten.

4.1 Einsatzmöglichkeiten

Das Dynamic Host Configuration-Protokoll (DHCP) wird in RFC 1541 spezifiziert. Es wurde entworfen, um Netzwerkknoten IP-Adressen und andere Netzwerkinformationen zuzuweisen, so dass sie ohne manuelle Konfiguration interagieren können. Bei einem IPv6-Netzwerk wird DHCP – wie im vorhergehenden Kapitel gezeigt – eigentlich nicht zum Konfigurieren von Adressen benötigt, allerdings müssen zum einen häufig weitere Konfigurationsinformationen übermittelt werden. Fast in allen Fällen wird es nötig sein, die Adresse eines DNS-Servers zu übertragen, und zum anderen kann auch einfach eine zustandsbehaftete Adressvergabe erwünscht sein, weil etwa die Anzahl der zu vergebenen Adressen (und damit die Anzahl der Netzwerkknoten) eingeschränkt werden soll. Ein drittes Einsatzszenario könnte auch die Verteilung von Präfixen für die zustandslose Autokonfiguration – die sogenannte Präfixdelegation – sein.

DHCP für IPv6 (DHCPv6) wurde in RFC 3315 spezifiziert und kann auf der einen Seite zustandsbehaftete Adressevergabe verwalten und davon unabhängig Konfigurationseinstellungen (etwa die Adresse des DNS-Servers) für IPv6-Hosts bereitstellen. Falls auch die Adressen über DHCP vergeben werden, spricht man von *zustandsbehaftetem DHCP*; werden hingegen nur die Konfigurationseinstellungen übertragen spricht man von *zustandsfreiem DHCP*.

4.1.1 Zustandsfreies DHCP

Bei zustandsfreiem DHCP werden keine Adressen vergeben. Dies wird stattdessen durch die Zustandslose Autokonfiguration mithilfe von Router-Ankündigungsnachrichten erreicht. Es können aber dennoch weitere Konfigurationsparameter übertragen werden. Dies beinhaltet etwa die Adressen von DNS- oder NTP (Network Time Protokoll) –Servern. Aber auch andere Konfigurationsparameter sind denkbar.

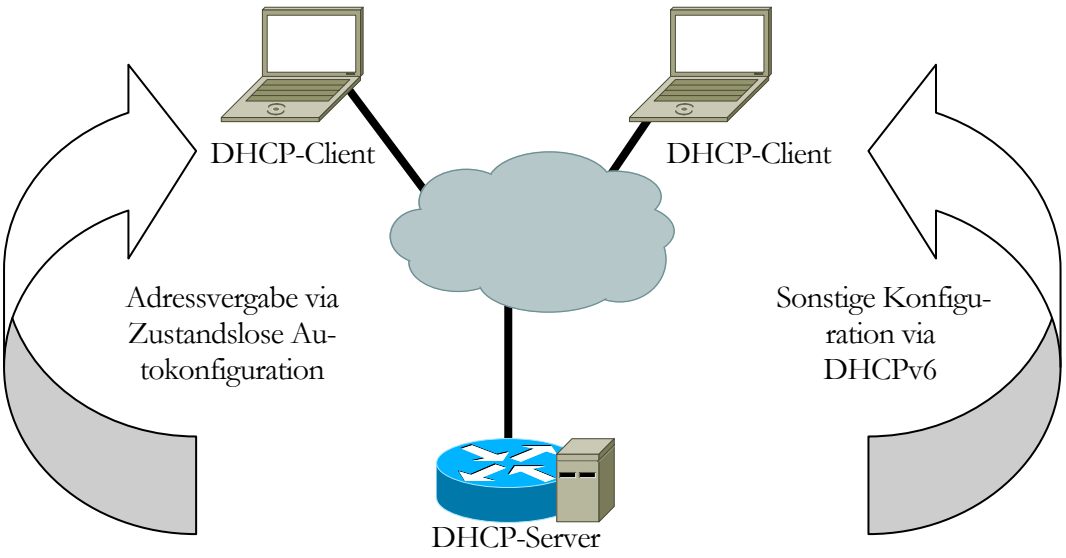

Abbildung 4.1 Skizze einer Topologie mit zustandsfreiem DHCP

4.1.2 Zustandsbehaftetes DHCP

Auch beim zustandsbehafteten DHCP werden Konfigurationsparameter übertragen, darüber hinaus werden aber auch Adressen an die Clients vergeben. Über die Adressen wird buchgeführt, dass heißt der Zustand des Servers unterscheidet sich nach einer Adressvergabe von dem vorherigen.

So ist es möglich, dass Hosts eine stabile Adresse erhalten oder dass nur eine gewisse Anzahl von Adressen vergeben wird.

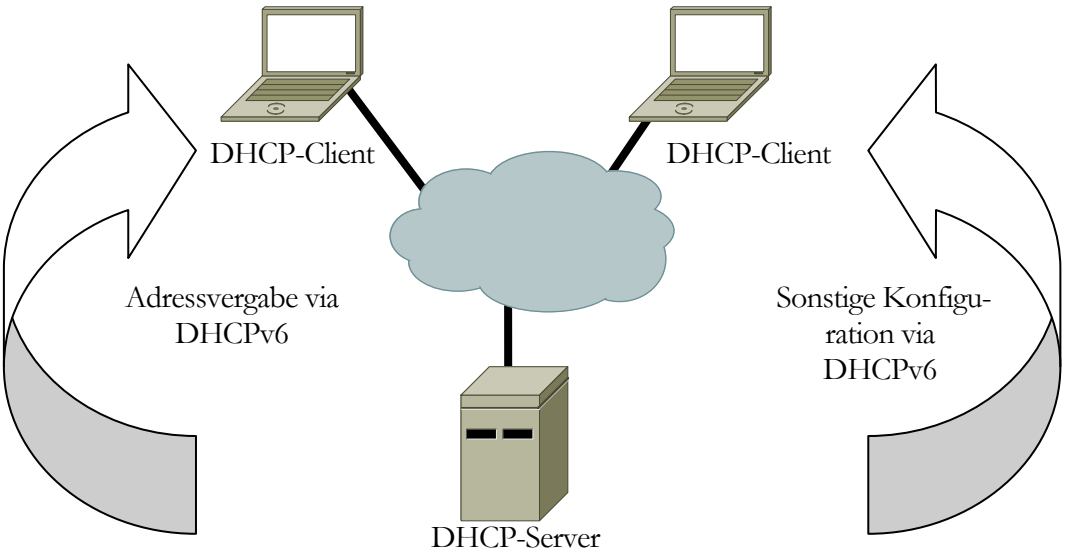

Abbildung 4.2 Skizze einer Topologie mit zustandsbehaftetem DHCP

4.1.3 Präfix-Delegation

Präfix-Delegation ist ein Spezialfall. Auf den Routern, die Adressen mittels zustandsloser Autokonfiguration Präfixe verteilen sollen, läuft dann ein DHCP-Client. Der Server vergibt in diesem Fall nicht einzelne Adressen sondern ganze Präfixe. Ein solches kann der Client-Router dann weiterverteilen. Die folgende Skizze verdeutlicht die Idee:

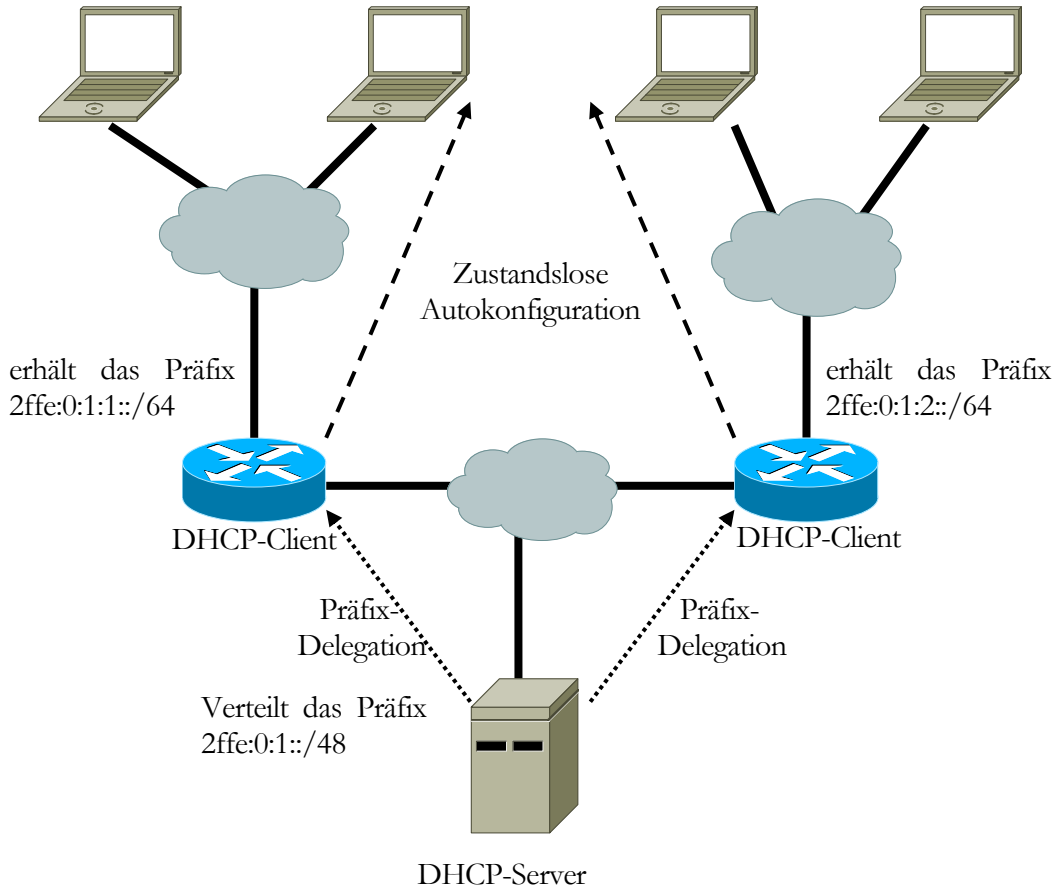

Abbildung 4.3 Skizze einer Topologie mit Präfix-Delegation

4.2 DHCPv6 und Router-Ankündigungen

Ob DHCPv6 für die automatische Adressvergabe und/oder den Empfang von anderen Konfigurationsinformationen verwendet wird, wird in der Router-Ankündigungs-Nachricht (siehe Abschnitt 3.3) festgelegt.

Durch Setzen des M-Schalters („verwaltete Adresskonfiguration“) wird ein Knoten dazu aufgefordert, ein DHCP zu verwenden, um zustandsbehaftete Adressen zu erhalten.

Durch Setzen des O-Schalters („andere zustandsbehaftete Konfiguration“) wird ein Knoten dazu aufgefordert, DHCP zu verwenden, um andere Konfigurationsparameter zu erhalten.

Wenn die Werte der M- und O-Schalter kombiniert werden, können folgende Fälle auftreten:

- Kein Schalter ist gesetzt: Diese Kombination entspricht einem Netzwerk ohne DHCPv6-Infrastruktur. Hosts verwenden Routerankündigungen für globale Adressen und andere Methoden (wie zum Beispiel manuelle Konfiguration), um andere Einstellungen zu konfigurieren.

- Beide Schalter sind gesetzt: DHCPv6 wird sowohl für Adressen als auch für andere Konfigurationseinstellungen verwendet. Diese Kombination entspricht dem zustandsbehaftetes DHCPv6.

- Nur der O-Schalter ist gesetzt: DHCPv6 wird nur dazu verwendet, andere Konfigurationseinstellungen, aber keine Adressen zuzuweisen. Diese Kombination entspricht dem zustandsloses DHCPv6.

- Nur der M-Schalter ist gesetzt: Diese unwahrscheinliche Kombination bestimmt, dass DHCPv6 für die Adresskonfiguration, nicht jedoch für andere Einstellungen, verwendet wird.

4.3 Komponenten

Wie DHCP für IPv4 enthält eine DHCPv6-Infrastruktur folgende Komponenten:

- DHCPv6-Clients

- DHCPv6-Server

- DHCPv6-Relay-Agenten, die Nachrichten zwischen Clients und Servern übermitteln, wenn sich Clients in Subnetzen ohne DHCPv6-Server befinden.

4.3.1 Clients

DHCPv6-Clients laufen im Regelfall auf Hostcomputern. Sie fordern Konfigurationsdaten von einem Server an.

4.3.2 Server

DHCPv6-Server laufen im Allgemeinen auf Routern. Sie liefern Konfigurationsdaten für die Clients. Beim zustandsbehafteten DHCP führen sie darüber hinaus über vorhandene Clients und die an sie vergebenen Adressen buch.

4.3.3 Relay-Agenten

DHCPv6-Relay-Agenten laufen immer auf Routern. Sie übermitteln Nachrichten zwischen Clients und Servern, wenn sich Clients in Subnetzen ohne Server befinden. Dieses Verfahren ist für den Client vollständig transparent. Er hat den Eindruck mit einem lokalen Server zu kommunizieren.

4.4 Kommunikationsweg

Die Konfiguration läuft über UDP, wie auch schon bei DHCP für IPv4. Allerdings wurden dort ausschließlich Broadcasts verwendet, die das Netz unnötig stark belasteten und unter IPv6 ohnehin nicht mehr zur Verfügung stehen. Deshalb wird stattdessen der folgende Kommunikationsweg eingesetzt:

Clients senden ihre Anfragen dabei in der Regel an die so genannte „Alle DHCP-Relay-Agents und -Server“ - Multicast-Adresse (FF02::1:2). Dies ist eine linklokale Multicast-Adresse und der zugehörigen Gruppe gehören alle Server und Relay-Agenten an. Diese wiederum lauschen auf dem Port 547. Es ist darüber hinaus möglich, dass der Client den Server per Unicast erreicht, allerdings müssen dafür sowohl Server als auch Client dieses Verhalten explizit erlauben. Ein solches Sonderverhalten wird in der Regel darüber hinaus nicht notwendig sein.

Lokale Server und Relay-Agenten erreichen einen Client mithilfe seiner linklokalen Unicastadresse, die mithilfe der zustandslosen Autokonfiguration generiert wurde. Clients lauschen dabei auf dem Port 546.

Die Kommunikation zwischen Server und Relay-Agent bietet die größte Freiheit. In beide Richtungen sind sowohl Uni- als auch Multicast möglich. Als Multicast-Adresse in Richtung Server wird dabei die site-lokale Multicast-Adresse „Alle DHCP Server“ (FF05::1:3) verwendet.

4.5 Der eindeutige DHCP-Bezeichner (DUID)

Bei DHCP für IPv4 wurde die MAC-Adresse verwendet, um einen beteiligten Knoten zu identifizieren. Dies birgt allerdings den Nachteil, dass bei Austausch der Netzwerkkarte auch eine neue Adresse bezogen werden muss, wohingegen es in bestimmten Anwendungsfällen wünschenswert sein kann, dieselbe Adresse ohne manuelle Eingriffe beibehalten zu können.

Bei DHCPv6 werden Knoten deshalb mithilfe des so genannten eindeutigen DHCP-Bezeichners (DUID, das Akronym steht für DHCP Unique Identifier) identifiziert. Dieser soll – falls es überhaupt möglich ist – stabil sein, sich also nicht verändern, selbst wenn Änderungen an der Netzwerk-Hardware vorgenommen werden. Unüblich ist, dass er, obwohl er quasi die eine Art Adresse bildet, in nur in einer Option einer DHCP-Nachricht übertragen wird. Dies begründet der RFC damit, dass zum einen seine Länge nicht vorgegeben sei und zum anderen er auch nicht in jedem Fall verwendet wird.

Der DUID kann - je nach Einsatzgebiet - auf verschiedene Arten generiert werden. Als Grund für diese Designentscheidung wird angeführt, dass er einfach zu generieren sein soll, und dass sich das, was einfach zu generieren sei, unter Umständen stark von einer Plattform zur nächsten unterscheide. So sind etwa Geräte ohne persistenten Speicher denkbar, die einen einmal generierter DUID nicht zwischenspeichern können und deshalb bei Wechsel der Netzwerkhardware einen anderen generieren müssen.

Im Moment werden drei verschiedene Verfahren zur Erstellung eines DUID spezifiziert, diese werden in der folgenden Tabelle miteinander verglichen.

Typ	**Basis für die Generierung**	**Voraussetzung**	**Bemerkung**
DUID-LLT	Sicherungsschichtadresse und Zeit	Beschreibbarer, persistenter Speicher	Kollision sehr unwahrscheinlich, DUID stabil (auch bei Hardwarewechsel)
DUID-EN	Eindeutiger Bezeichner, der vom Hersteller vergeben	Ein solche vom Hersteller vergebe-	

	wird	ner Bezeichner	
DUID-LL	Nur Sicherungsschichtadresse	keine	Netzwerkadapter muss permanent mit dem Gerät verbunden sein

Der RFC empfiehlt, wann immer möglich DUID-LLT zu verwenden, denn mit diesem Verfahren wird zum einen eine stabile DUID erreicht, die sich selbst beim Wechsel der Netzwerkhardware-Wechsel nicht verändert, zum anderen wird selbst in dem Fall, dass ein Netzwerkadapter aus einem Gerät entfernt und in ein anderes eingebaut wird, keine Kollisionen auf, da eine neue DUID berechnet wird. Sollte es nicht möglich sein, empfiehlt der RFC, DUID-LL zu verwenden. In diesem Fall muss allerdings sichergestellt werden, dass der Netzwerkadapter niemals an einem anderen Gerät betrieben wird.

4.6 Die Identitätszuordnung (IA)

Bei DHCP für IPv4 wird jeder Schnittstelle genau eine Adresse zugewiesen. Deshalb ist es dort nicht nötig, diese Zuweisungen zu gruppieren. Bei IPv6 werden jedoch häufig mehrere Adressen für eine Schnittstelle angefordert (etwa für die verschiedenen Rollen die das Gerät im Netzwerk spielt) und deshalb ist es wünschenswert, diese auch Gruppieren zu können. Hier kommt die Identitätszuordnung (IA, das Akronym steht für Identity Association) ins Spiel. Diese weißt einer Schnittstelle verschiedene Konfigurationsparameter zu. Dies könnten auch eine oder mehrere Adressen sein.

Ein IAID (IA-Bezeichner/IA-Identifier) bezeichnet dabei zusammen mit der DUID genau eine Schnittstelle eines Hosts. Auf der anderen Seite muss der Client jeder seiner Schnittstellen, die per DHCPv6 konfiguriert werden sollen, mindestens eine IAID zuordnen. Dabei sollen diese pro Client eindeutig sein und sich möglichst nicht nach einem Neustart verändern. Dies könnte entweder durch Speichern der IAIDs ermöglicht werden oder durch einen Algorithmus, der immer die gleichen IAIDs vergibt, solange sich die Netzwerkhardware-Konfiguration des Clients nicht verändert.

4.7 Nachrichtenformat

4.7.1 Kommunikation zwischen Client und Server/Relay-Agent

Die folgende Abbildung zeigt den grundlegenden Aufbau eines DHCPv6-Paketes, wie es zwischen Client und Server verwendet wird. Dabei fällt insbesondere die Variabilität auf. Nur zwei Felder (der Nachrichten-Typ (siehe Abschnitt 4.7.3a+b) und die Transaktionsnummer (siehe Abschnitt 4.7.4)) sind Pflicht, alle anderen Informationen werden in Form von Optionen (siehe Abschnitt 4.7.5) übertragen.

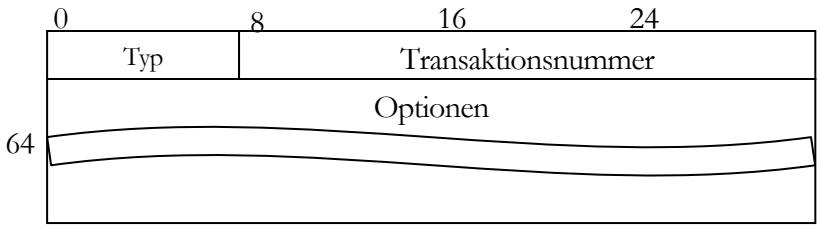

Abbildung 4.4 Das DHCPv6-Paket (Client und Server)

4.7.2 Kommunikation zwischen Relay-Agent und Server/Relay-Agent

Falls an der Kommunikation ein Relay-Agent beteiligt ist und eine Paket verpackt werden muss, kommt ein zweiter, etwas komplexerer Paket-Typ zum Einsatz. Die folgende Abbildung illustriert ihn.

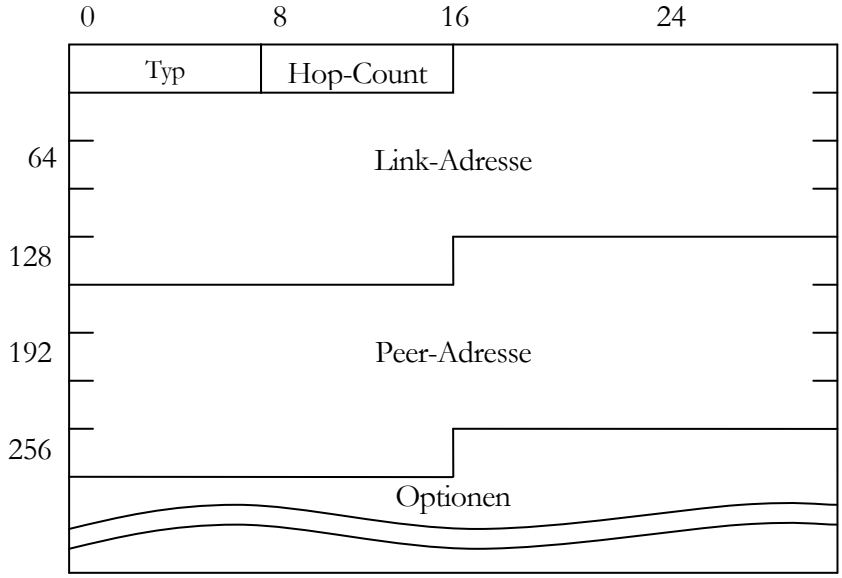

Abbildung 4.5 Das DHCPv6 Paket (Relay-Agent)

Die folgende Tabelle beschreibt die einzelnen Felder:

Feld	**Beschreibung**
Typ	Dieses Feld muss RELAY-FORW oder RELAY-REPL sein (siehe Abschnitt 4.7.3c)
Hop-Count	Dieses Feld bezeichnet die Anzahl der Relay-Agenten, von denen das Paket weitergeleitet wurde (RELAY-FORW) oder noch weitergeleitet werden muss (RELAY-REPL)
Link-Adresse	Die Link-Adresse ist eine IPv6-Adresse, die sowohl vom Server als auch vom zugehörigen Relay-Agenten übereinstimmend konfiguriert sein muss. Sie bezeichnet das Subnetz, an dem der an der Kommunikation beteiligte Client angeschlossen ist. Dabei kann es sich etwa um das Präfix für das entsprechende Netz handeln.
Peer-Adresse	Diese Adresse bezeichnet den Absender der eingebetteten Nachricht.

Optionen	In den Optionen, die eine variable Länge haben, wird die eigentliche Nachricht eingeschlossen. Dazu muss in einer solchen Nachricht „Relay-Nachricht"-Option enthalten sein.

4.7.3 Nachrichten-Typen

Die folgenden Nachrichtentypen sind dabei vorgesehen:

a) Vom Client an den Server/Relay-Agent

Typ	**Beschreibung**
SOLICIT	Der Client sendet eine solche Nachricht, um Server zu finden
REQUEST	Der Client fordert mit diesem Nachrichtentyp Konfigurationsparameter von einem bestimmten Server an.
CONFIRM	Der Client sendet eine solche Nachricht an alle verfügbaren Server, um zu überprüfen, ob die gewählte Adresse noch auf dem entsprechenden Link verfügbar ist.
RENEW	Der Client sendet eine Nachricht dieses Typs, zu dem Server, der ihm seine aktuelle Adresse ursprünglich zugeteilt hat. Darin fordert er ihn auf, die Lebenszeit dieser Adresse zu verlängern.
REBIND	Der Client sendet eine solche Nachricht an alle verfügbaren Server, um die Lebenszeit seiner aktuellen Adressen verlängern zu lassen. REBIND-Nachrichten werden versendet, wenn auf eine RENEW-Nachricht keine Antwort einging.

RELEASE	Der Client teilt dem Server, der ihm seine Adresse zugewiesen hat, mit diesem Nachrichtentyp mit, dass er diese Adresse nicht mehr verwenden wird.
DECLINE	Der Client sendet eine solche Nachricht, um mitzuteilen, dass eine oder mehrere ihm vom Server zugewiesene Adressen, auf dem Link bereits verwendet werden.
INFORMATION-REQUEST	Der Client sendet eine solche Nachricht an einen Server, um Konfigurationsparameter ohne Zuweisung einer Adresse zu erhalten.

b) Vom Server/Relay-Agent an den Client

Typ	**Beschreibung**
ADVERTISE	Der Server antwortet mit einer solchen Nachricht auf ein SOLICIT und gibt damit an, dass er DHCP-Dienste anbietet.
REPLY	Der Server sendet REPLY-Nachrichten als Antwort auf fast alle Anfragen des Clients.
RECONFIGURE	Ein Server schickt einem Client eine solche Nachricht, um ihm mitzuteilen, dass neue Konfigurationsparameter vorliegen, und dass er eine INFORMATION-REQUEST bzw. RENEW/REBIND-Nachricht schicken soll, um diese zu erhalten.

c) Zwischen Server und Relay-Agent

Typ	Beschreibung
RELAY-FORW	Der Relay-Agent verschickt eine solche Nachricht, um Nachrichten an Server weiterzuleiten. Dazu kann die Nachricht entweder direkt an den Server oder an einen anderen Relay-Agenten übermittlelt werden. Die eingebettete Nachricht, bei der es sich entweder um eine Client-Nachricht oder eine RELAY-FORW-Nachricht handelt, wird in einer Option übertragen.
RELAY-REPL	Ein Server schickt eine solche Nachricht an einen Relay-Agenten, um eine Nachricht an einen Server weiterleiten zu lassen. Dabei kann es vorkommen, dass die Nachricht abermals in eine RELAY-REPL-Nachricht eingebetet wird, um sie auf dem Weg zum Client über weitere Relay-Agenten zu leiten.

4.7.4 Transaktionsnummer

Transaktionsnummern werden verwendet, um Clientanfragen und Serverantworten miteinander zu synchronisieren, das bedeutet, dass ein Nachrichtenpaar zwischen Clientanfragen und Serverantworten dieselbe Transaktionsnummer verwendet. Außerdem soll sich die Transaktionsnummer bei Neuübertragungen von Anfragen nicht verändern, damit, falls nur die Antwort verloren gegangen ist, dies vom Server auch richtig gedeutet werden kann.

4.7.5 Optionen

In den Optionen werden fast sämtliche Informationen übertragen. Eine Nachricht kann beliebig viele Optionen enthalten, die alle das gleiche Format haben. Dieses wird in der folgenden Abbildung gezeigt:

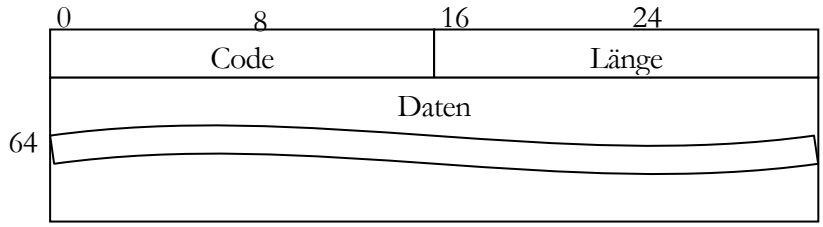

Abbildung 4.6 Aufbau einer Option

Die Bedeutung der einzelnen Felder wird in der folgenden Tabelle erläutert:

Feld	Länge	Beschreibung
Code	8 Bit	Dieses Feld beschreibt den Typ der Option
Länge	8 Bit	Dieses Feld beschreibt die Länge der Optionsdaten in Byte
Daten	variabel	Daten der Option, Inhalt ist abhängig vom Options-Code

Dies sind die wichtigsten Optionen, die in RFC 3315 spezifiziert wurden:

Option	Beschreibung
Client-Bezeichner-Option	Eine solche Option enthält den DUID des Clients, der an der Kommunikation beteiligt ist. Sendet der Client eine Nachricht mit dieser Option, ist es also eine Art Absender-Adresse. Sendet der Server eine solche Option, enthält sie den Empfänger.
Server-Bezeichner-Option	Diese Option enthält den DUID des Servers. Je nachdem, wer die Nachricht sendet, kann sie sowohl den Absender als auch den Empfänger kennzeichnen.
IA_NA	Eine solche Option bezeichnet a) den Wunsch, eine IA für nicht-temporäre Adressen zu erhalten (falls sie der Client sendet) oder b)

Option	Beschreibung
	die Zuordnung einer solchen IA (falls sie der Server sendet) Sie enthält die IAID, zwei Gültigkeitsdauern (T1 – Zeit bis zum nächsten Renew, T2 – Zeit bis zum nächsten Rebind), sowie eine oder mehrere IA-Adress-Optionen. Sendet der Client diese Daten, sind sie als seine bevorzugte Werte zu verstehen, sendet sie hingegen der Server sind sie tatsächlich zugeordnet.
IA_TA	Eine solche Option bezeichnet a) den Wunsch, eine IA für temporäre Adressen (siehe Abschnitt 3.5) zu erhalten (falls sie der Client sendet) oder b) die Zuordnung einer solchen IA (falls sie der Server sendet) Sie enthält wie die IA_NA-Option die IAID und eine oder mehrere IA-Adress-Optionen, aber keine Gültigkeitsdauern (temporäre Adressen können nicht verlängert werden) Sendet der Client diese Daten, sind sie als seine bevorzugte Werte zu verstehen, sendet sie hingegen der Server sind sie tatsächlich zugeordnet.
Präferenz-Option	Diese Option bezeichnet die Präferenz eines Servers. Wenn der Client mehrere Server zur Auswahl hat, wird er mit demjenigen mit der höchsten Präferenz weiter kommunizieren.
Relay-Nachricht-Option	Diese Option wird in RELAY-FORW- und RELAY-REPL-Nachrichten benötigt. Sie enthält die eingekapselte DHCP-Relay-Nachricht.
Rapid-Commit-Option	Diese Option kennzeichnet den Wunsch des Clients, die 2-Wege-Kommunikation statt dem Standard (der 4-Wege-Kommuniktion) einzusetzen. (Siehe Abschnitt 4.8)

IA-Optionen

Option	Beschreibung
IA-Adress-Option	Eine solche Option enthält zum einen eine IPv6-Adresse und zum anderen eine bevorzugte und gültige Lebenszeit für diese Adresse. Sendet ein Client eine solche Option, kennzeichnet sie seine bevorzugten Werte für diese Parameter. Sollte ihm die Adresse egal sein, trägt er dort die unspezifizierte Adresse (::) ein. Sendet der Server hingegen eine solche Option, kennzeichnet sie eine tatsächlich zugeordnete Adresse.

4.8 Kommunikation bei zustandsbehaftetem DHCP

Die Kommunikation im zustandsbehafteten Fall zwischen Client und Server lässt sich in die folgenden Fälle aufteilen:

- Die Kommunikation bei der Adresszuordnung,
- der Versuch, sich eine bestehende Adresse bestätigen zu lassen, wenn der Client das Subnetz gewechselt haben könnte
- das Verhalten bei Ablauf der Lebenszeiten der Adressen,
- die Rückgabe der Adressen und
- die Behandlung der Fehlerfälle

Diese Fälle werden im Folgenden erläutert.

4.8.1 Kommunikation bei der Adresszuordnung

DHCPv6 bietet zwei Verfahren an, wie die initiale Kommunikation zur Adresszuordnung aussehen kann. Die eine benötigt 4 Nachrichten, die andere kommt mit 2 Nachrichten aus.

4.8.1.1 4-Wege-Kommunikation

Standardmäßig besteht die initiale Kommunikation zwischen Client und Server aus den 4-Nachrichten: SOLICIT, ADVERTISE, REQUEST, REPLY. Diese Kommunikationsform wird deshalb auch als 4-Wege-Kommunikation bezeichnet. Die folgende Skizze zeigt das Schema der Kommunikation:

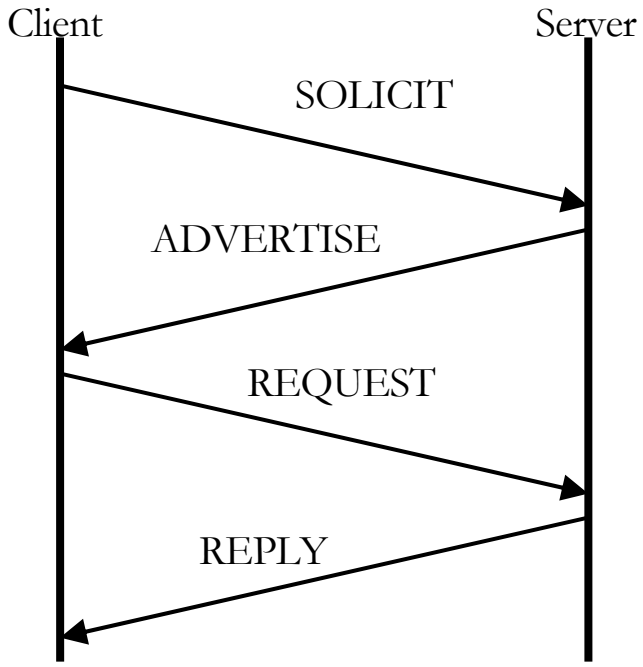

Abbildung 4.7 Schema: 4-Wege-Kommunikation

Der Client beginnt damit, eine SOLICIT-Nachricht an die „Alle DHCP-Relay-Agenten und Server"-Multicast-Adresse zu senden. Darin enthalten ist eine Client-Bezeichner-Option, die ihn identifiziert, sowie IA_NA oder IA_TA-Optionen, die beschreiben, welche Adressen er zu welchen Konditionen gerne hätte.

Alle verfügbaren Server antworten darauf mit ADVERTISE-Nachrichten, die entsprechende Adressen (IA_NA bzw. IA_TA-Optionen) enthalten. Dabei müssen die tatsächlichen Werte nicht mit den

Wünschen des Clients übereinstimmen. Diese Nachricht enthält darüber hinaus einen Präferenzwert des Servers.

Der Client wählt nun einen der Server aus, dabei spielt der Präferenzwert die größte Rolle, allerdings darf der Client auch einen Server mit niedrigerem Präferenzwert auswählen, wenn ihm dessen Angebot besser gefällt. Er sendet nun eine REQUEST-Nachricht an die „Alle DHCP-Relay-Agenten und Server"-Multicast-Adresse. Diese enthält eine Server-Bezeichner-Option, die den gewünschten Server festlegt. Dieser wiederum antwortet nun noch mit einer REPLY-Nachricht, die die Daten noch einmal bestätigt und ggf. noch zusätzliche Konfigurationsparameter enthält. Der Client führt nun noch eine DAD durch, und nimmt die ihm zugewiesenen Adressen dann an.

Dieses Kommunikationsschema hat den Nachteil, dass im häufigsten Fall (es gibt nur einen DHCPv6-Server im Subnetz) eigentlich zu viele Nachrichten ausgetauscht werden. Deshalb spezifiziert der RFC darüber hinaus noch eine weitere Kommunikationsform.

4.8.1.2 2-Wege-Kommunikation

Die 2-Wege-Kommunikation besteht aus nur zwei Nachrichten. Einer SOLICIT und einer REPLY-Nachricht. Die folgende Skizze erläutert das Schema:

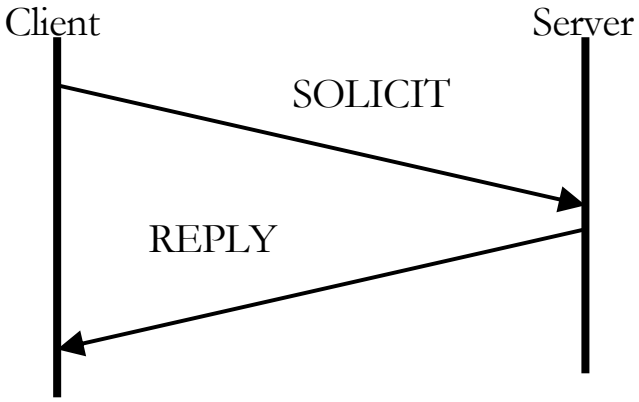

Abbildung 4.8 Schema: 2-Wege-Kommunikation

Die SOLICIT-Nachricht des Clients enthält nun neben der Client-Bezeichner-Option und den IA-Optionen noch die Rapid-Commit-Option, die besagt, dass der Client die 2-Wege-Option enthält.

Ein Server, der eine solche Nachricht erhält antwortet

- mit einer REPLY-Nachricht mit ebenfalls angefügter Rapid-Commit-Option, falls er die 2-Wege-Kommunikation unterstützt,
- mit einer ADVERTISE-Nachricht, falls er die 2-Wege-Kommunikation nicht unterstützt.

Falls ein Client eine REPLY-Nachricht mit Rapid-Commit-Option erhält, nimmt er die darin enthaltenen Daten an (und führt eine DAD durch)

Falls er keine solche Nachricht, wohl aber eine ADVERTISE-Nachricht erhält, fährt er wie bei der 4-Wege-Kommunikation fort.

Diese Kommunikationsform hat den folgenden Nachteil: Falls mehrere Server lokal verfügbar sind, werden alle auf das SOLICIT mit einem REPLY antworten. Der Client wird allerdings nur eines dieser Angebote annehmen, seine Entscheidung aber nicht mehr mitteilen. D.h. die Server haben Adressen vergeben, die eigentlich gar nicht verwendet werden. Dieses Problem könnte man auf zwei Arten umgehen:

- Man könnte nur einen Server je Subnetz so konfigurieren, dass er die 2-Wege-Kommunikation unterstützt. Damit kann dieser Server praktisch sicher sein, dass sein Angebot angenommen wurde.
- Man könnte die Lebenszeiten (siehe Abschnitt 4.8.3) sehr kurz wählen, so dass nicht verwendete Adressen schnell wieder für eine weitere Vergabe verfügbar werden.

4.8.2 Bestätigung von bestehenden Adressen

Es kann mehrere Fälle geben, in denen ein Client nicht sicher sein kann, ob er sich noch im selben Subnetz befindet, wie zu dem Zeitpunkt, als ihm ein DHCP-Server eine Adresse zugeordnet hat. Dazu zählen etwa ein Neustart des Clients, ein Wechsel aus dem Stand-by-Zustand oder der Wechsel des Access-Points im Falle einer drahtlosen Netzwerkverbindung.

Wann immer der Client sich nicht mehr sicher sein kann, dass er noch mit demselben Subnetz verbunden ist, muss er eine CONFIRM-Nachricht an denjenigen Server schicken, der ihm seine Adressen zugeordnet hat, damit dieser ihm in einer REPLY-Nachricht diese Adressen bestätigt.

4.8.3 Ablauf der Lebenszeiten

Jeder nicht-temporären Adresse, die per DHCPv6 vergeben wurde, sind insgesamt 4 Lebenszeiten zugeordnet. Neben der bevorzugten und der gültigen Lebenszeit (siehe Abschnitt 3.4) kommen nun noch die sogenannte RENEW-Zeit (auch T1-Zeit) und die REBIND-Zeit (auch T2-Zeit) für die IA hinzu.

4.8.3.1 Ablauf der RENEW-Zeit

Auf den Ablauf dieser Zeit reagiert der Client mit dem Versenden einer RENEW-Nachricht, mit der er versucht, sich seine Adresse von dem Server bestätigen zu lassen, der sie ihm auch zugeteilt hat. Diese Nachricht wird zwar an die „Alle DHCP-Relay-Agenten und Server"-Multicast-Adresse gesendet, enthält aber den Server-Bezeichner desjenigen Servers, der ihm seine Adresse zugeordnet hat. In der Regel wird also auch nur dieser (mit einer REPLY-Nachricht) antworten, dennoch könnten (etwa bei bekannten Ausfall dieses Servers) auch auf diese Nachricht schon redundante Server antworten.

4.8.3.2 Ablauf der REBIND-Zeit

Sollte der Client keine Antwort auf sein RENEW bekommen und nun auch noch die T2-Zeit abgelaufen sein, versucht er sich seine Adresse von einem beliebigen Server bestätigen zu lassen. Dazu sendet er eine REBIND-Nachricht an die „Alle DHCP-Relay-Agenten und Server"-Multicast-Adresse. Wenn ein Server ihm diese Adresse bestätigen kann, antwortet er mit einer REPLY-Nachricht.

4.8.3.3 Ablauf der bevorzugten Lebenszeit

Sollte auch auf das REBIND keine Antwort gekommen sein und nun auch die bevorzugte Lebenszeit abgelaufen sein, wird der Client versuchen, sich eine neue Adresse mittels der 4-Wege- bzw. der 2-Wege-Kommunikation zu beschaffen. Die alte Adresse gilt nunmehr als veraltet und darf nur noch verwendet werden um bestehende Kommunikationen zu Ende zu bringen.

4.8.3.4 Ablauf der gültigen Lebenszeit

Sollte auch die gültige Lebenszeit abgelaufen sein, darf die Adresse gar nicht mehr verwendet werden. Der Client wird weiterhin versuchen mittels einer SOLICIT-Nachricht lokale Server zu finden um sich neu konfigurien zu können.

4.8.4 Rückgabe von Adressen

Wenn ein Client die ihm zugewiesenen Adressen zurückgeben möchte – etwa weil er sich endgültig aus einem Netz abmelden oder neue Adressen beziehen möchte, sendet er eine RELEASE-Nachricht an die „Alle DHCP-Relay-Agent und Server"-Multicast-Adresse. Diese Nachricht muss seinen eigenen DUID und den des Servers enthalten, der ihm die Adressen zugeordnet hat. Der Server antwortet mit einer REPLY-Nachricht, in der er die Rückgabe bestätigt.

4.8.5 Fehlerfälle

Es können zwei grundsätzlich verschiedene Fehlerfälle auftreten. Zum einen könnte eine Nachricht verloren gehen und zum anderen könnte eine zugewiesene Adresse bereits im Netzwerk vergeben worden sein. Die Reaktion auf diese Fehler wird im Folgenden beschrieben.

4.8.5.1 Nachrichtenverlust

Der Client ist für die Verlässlichkeit sämtlicher von ihm initiierten Nachrichtenwechsel verantwortlich. Er muss also nicht bestätigte Anfragen wiederholen. Dabei muss er die Transaktionsnummer der ursprünglichen Nachricht wiederholen, um sicherzustellen, dass – falls nur die Antwortnachricht verloren gegangen ist – der Server diese wiederholte Anfrage von einer gänzlich neuen unterscheiden kann.

4.8.5.2 Ablehnen einer Adresse

Falls der Client bei der DAD feststellt, dass eine ihm zugewiesene Adresse bereits verwendet wird, so darf er diese natürlich nicht verwenden. Darüber hinaus muss er den Server von diesem Fehlerfall in Kenntnis setzen, indem er eine DECLINE-Nachricht schickt mit der Angabe der abgelehnten Adressen in entsprechenden IA-Optionen. Der Server markiert die entsprechenden Adressen als benutzt (damit er sie nicht noch einmal vergibt) und löscht die zu den Adressen gehörigen IAs. Dann bestätigt er den Vorgang mit einer REPLY-Nachricht. Der Client kann darauf einen neuen Versuch beginnen, mittels der 2- oder 4-Wegekommunikation Adressen zu beziehen.

4.9 Kommunikation bei zustandslosem DHCP

Bei zustandslosem DHCP können neben der Fehlerbehandlung bei verlorenen Nachrichten nur zwei Kommunikationsfälle auftreten.

Diese beiden Fälle sind:

- Die Informationsanforderung durch den Client und
- Aufforderung zu einer solchen durch den Server

Diese beiden Fälle werden im Folgenden erörtert.

4.9.1 Informationsanforderung

Möchte ein Client keine Adressen, wohl aber sonstige Konfigurationsparameter erhalten, sendet er eine INFORMATION-REQUEST-Nachricht an die „Alle DHCP-Relay-Agenten und Server"-Multicast-Adresse. Jeder Server, der eine solche Nachricht erhält, antwortet darauf mit einer REPLY-Nachricht, die die gewünschten Konfigurationsparameter enthält. Falls der Client mehrere Antworten erhält, wählt er eine (oder mehrere) davon aus und konfiguriert sich mit den entsprechenden Werten.

4.9.2 Rekonfigurationsaufforderung

Der Client kann entweder in der INFORMATION-REQUEST-Nachricht (im Falle von zustandslosem DHCP) oder in der SOLICIT-Nachricht (im Falle von zustandsbehaftetem DHCP) angeben, dass er bereit ist, sich zu einem späteren Zeitpunkt dazu auffordern zu lassen, neue Konfigurationsparameter zu erhalten. In diesem Falle sendet ihm der Server, sobald neue oder aktualisierte Konfigurationsparameter zur Verfügung stehen eine RECONFIGURE-Nachricht. Der Client antwortet darauf mit einer INFORMATION-REQUEST- oder RENEW-Nachricht, so dass die neuen Parameter bezogen werden können.

4.10 Präfix-Delegation

Neben den bisher beschriebenen Anwendungsfällen wurde zu Beginn des Kapitels noch die Präfix-Delegation erwähnt. Diese ist in RFC 3633 spezifiziert.

Dabei läuft auf den einzelnen Routern ein DHCP-Client, dieser empfängt aber vom Server keine einzelnen Adressen, sondern ganze Präfixe. Dafür wurde im oben genannten RFC die weitere Option namens IA_PD spezifiziert, die ganze Präfixe inklusive deren Präfixlänge sowie ihrer Renew- und Rebindzeiten tragen kann. Darüber hinaus kann der DHCP-Server auch die Lebenszeiten der aus dem Präfix generierten Adressen festlegen.

Der empfangende Router kann nun dieses Präfix mittels zustandsfreier Autokonfiguration weiterverteilen.

Anwendungsgebiete dieses Verfahrens sind Netze, deren Adresse sich häufig verändert, da das Renumbering, also die Änderung der Adressen so stark erleichtert werden kann. Sollten allerdings zusätzliche Konfigurationsparameter benötigt werden, müssen diese auf einem anderen Weg verteilt werden, so dass sich dann gleich das im Anschluss erörterte Verfahren mit Relay-Agenten anbietet.

4.11 Relay-Agenten

Relay-Agenten leiten Nachrichten zwischen Clients und Server weiter, wenn kein Server im lokalen Subnetz verfügbar ist. Dies hat vor allem den Vorteil, dass nur ein Server konfiguriert werden muss, und dass somit nur an einem Punkt Änderungen anstehen, wenn beispielsweise ein Providerwechsel mit etwaigen neuer Adressvergabe ansteht. Die zentrale Administration hat auch hier den Vorteil, dass sie weniger Fehleranfällig ist, sowie schneller und einfacher durchgeführt werden kann.

Relay-Agenten werden dabei nur insofern konfiguriert, dass sie wissen müssen, auf welchen ihrer Schnittstellen sie mit Clients rechnen müssen, so dass sie dort auf die entsprechende Multicast-Adresse lauschen und Nachrichten dorthin weiterleiten können und wie den (oder die) DHCP-Server erreichen können. Außerdem muss jedes Subnetz, auf dem mit Clients kommuniziert wird, einen eindeutigen Bezeichner bekommen, damit der DHCP-Server, der eine weitergeleitete Nachricht erhält, diese auch richtig zuordnen kann.

Wenn der Relay-Agent nun eine Nachricht von einem Client erhält, packt er diese in eine RELAY-FORW-Nachricht ein, die auch den Bezeichner des Subnetzes, an das der Client angeschlossen ist, enthält. Nun sendet er diese Nachricht an den (oder die) konfigurierten DHCP-Server.

Diese generieren wie gewohnt eine Antwort und packen diese in eine RELAY-FORW-Nachricht ein, die wiederum auch den Subnetzbezeichner enthält und senden diese Nachricht an einen Relay-Agenten, der mit diesem Subnetz verbunden ist. Dieser Pakt das Paket aus und sendet den Inhalt, also die Antwort des Servers wie gewohnt an den gewünschten Client.

Für den Client ist dieses Verfahren vollständig transparent.

5 Alternative Vorschläge

Die beiden bisher gezeigten Lösungsansätze haben beide große Nachteile.

Der Hauptnachteil der zustandslosen Autokonfiguration ist, dass sie in den meisten Fällen nicht ausreicht, um einen Hostrechner vollständig zu konfigurieren. In der überwiegenden Mehrzahl der Fälle wird mindestens noch die Adresse eines DNS-Servers benötigt, um den Rechner im Internet sinnvoll einsetzen zu können. Dies stellt einen gravierenden Mangel dar, dessen auftreten bei dem ansonsten sehr durchdachten Konzept der Zustandslosen Autokonfiguration sehr überrascht.

Natürlich kann man dies mit dem im vorhergehenden Kapitel beschriebenen Verfahren des zustandsfreien DHCP umgehen, allerdings benötigt man dazu eine zusätzliche Client- und Server-Infrastruktur, die nur für die Übermittlung einer einzigen Adresse übertrieben wirkt.

Dabei wäre es ohne größeren Aufwand möglich gewesen, auch ohne zustandsfreies DHCP an die Konfigurationsdaten zu kommen. Drei Vorschläge um dies zu erreichen werden im Folgenden erörtert.

5.1 Erweiterung der Router-Ankündigungsnachrichten

Nahe liegend wäre es zum einen, die Router-Ankündigungsnachrichten geeignet zu erweitern. (Zum Aufbau dieser Nachrichten siehe Abschnitt 3.3). Dazu böte sich etwa eine Option namens „DNS-Server“-Option an, die die Adresse des nächsten DNS-Servers enthalten könnte.

Dazu müsste diese natürlich auf dem Router konfiguriert werden, was allerdings aufgrund des ohnehin vorhandenen Konfigurationsaufwandes mit verfügbaren Präfixen kein größeres Problem darstellen würden. Ferner müsste der Dienst für die Router-Ankündigungen entsprechend erweitert werden. RFC 2461 gibt zudem an, dass neue Versionen dieser Nachrichten auch zusätzliche Optionen enthalten könnten. Laut dieser Spezifikation müssen alle bestehenden Implementationen des Protokolls un-

bekannte Optionen stillschweigend ignorieren. Also wäre eine Erweiterung des Protokolls an dieser stelle ohne Probleme möglich.

Clients müssten natürlich insofern modifiziert werden, dass sie die neue Option auslesen und entsprechend Verarbeiten. Aber auch dies würde die Arbeitsweise des Clients in keiner Weise beeinträchtigen.

Es bleibt also festzustellen, dass eine solche Erweiterung des ICMPv6-Protokolls zulässig wäre und keine zusätzlichen Probleme schaffen würde.

5.2 Vergabe einer DHCP-Server-Anycast-Adresse

Der zweite Vorschlag kommt sogar ganz ohne Modifikation der Software aus. Es müsste lediglich von der IANA eine globale DNS-Anycast-Adresse vergeben würden, mit der alle Clients vorkonfiguriert werden könnten. Router könnten dann beispielsweise mithilfe von NAT (das ja prinzipiell auch unter IPv6 funktioniert) diese Adresse in eine gültige Unicast-Adresse eines DNS-Servers übersetzen.

In diesem Fall, muss lediglich der Router mit der DNS-Server-Adresse konfiguriert werden. Auch dieses Verfahren scheint also konsistent mit der bisherigen Spezifikation umsetzbar zu sein. Warum das aufwendige Berfahren des Zustandslosen DHCP sich also scheinbar durchgesetzt hat, bleibt rätselhaft.

5.3 Verwendung des Default-Routers als DNS-Server

Als dritte Möglichkeit würde es sich anbieten, den Client so zu modifizieren, dass der Default-Router standardmäßig auch als DNS-Server angenommen wird, solange kein anderer DNS-Server erreichbar ist. Auf dem Router müsste dann allerdings ein DNS-Server installiert sein, der Namensauflösungen rekursiv auflöst.

6 Praxis

In diesem Kapitel soll das in den vorhergehenden Kapiteln in der Theorie Erörterte nun in der Praxis demonstriert werden. Dabei wird die Virtualisierungsumgebung VNUML zum Einsatz kommen, in die im folgenden Abschnitt kurz eingegangen wird. Danach folgen die praktische Demonstration der zustandslosen Autokonfiguration und von DHCPv6. Dabei wird sowohl auf die Installation, als auch die Konfiguration und die Benutzung der dafür benötigten Dienste eingegangen und es werden einige Beispiele gegeben.

6.1 Einführung

6.1.1 VNUML

VNUML (Virtual Network User Mode Linux) ist ein an der Technischen Universität Madrid entwickeltes System, das es erlaubt mithilfe von recht einfachen XML-Dateien selbst komplexe Netzwerkszenarien zu modellieren und diese dann als Simulation laufen zu lassen. Die einzelnen Gastsysteme basieren dabei auf UML (also User Mode Linux), was es erlaubt, vollwertige Linuxsysteme zu virtualisieren. So kann zum Beispiel neue oder unbekannte Software auf einfache und günstige Weise in einem virtuellen Netzwerk getestet werden und besonders für die Lehre und für Übungszwecke bietet sich dieses System sehr an. Für weitere Informationen über VNUML und dessen Benutzung sei an dieser Stelle die Homepage[2] der Universität Koblenz-Landau zu diesem Thema empfohlen.

[2] http://www.uni-koblenz.de/~vnuml/index.de.php

6.1.2 IPv6 unter VNUML

Solange das verwendete Gastsystem IPv6 unterstützt, kann dieses Protokoll auch unter VNUML vollwertig verwendbar. Als Einführung sei [IPV6VNUML] empfohlen.

6.2 Zustandslose Autokonfiguration linklokaler Adressen

Zunächst einmal werden wir die automatische Konfiguration der linklokalen Adressen und die *duplicate adress cdetection* betrachten.

Abbildung 6.1 zeigt ein einfaches Szenario mit 3 Host-Computern und einem Router, der später bei der Konfiguration der globalen Adressen zum Einsatz kommen wird. In der Abbildung sind die MAC-Adressen der einzelnen Schnittstellen angegeben, da die IPv6-Adressen der Host-Computer automatisch konfiguriert werden sollen. Die Konfigurations-Datei für dieses Szenario ist im Anhang im Abschnitt 8.1.1 zu finden. Wir werden zunächst den Fall betrachten, dass die Autokonfiguration gelingt, und daraufhin einen Misserfolg provozieren.

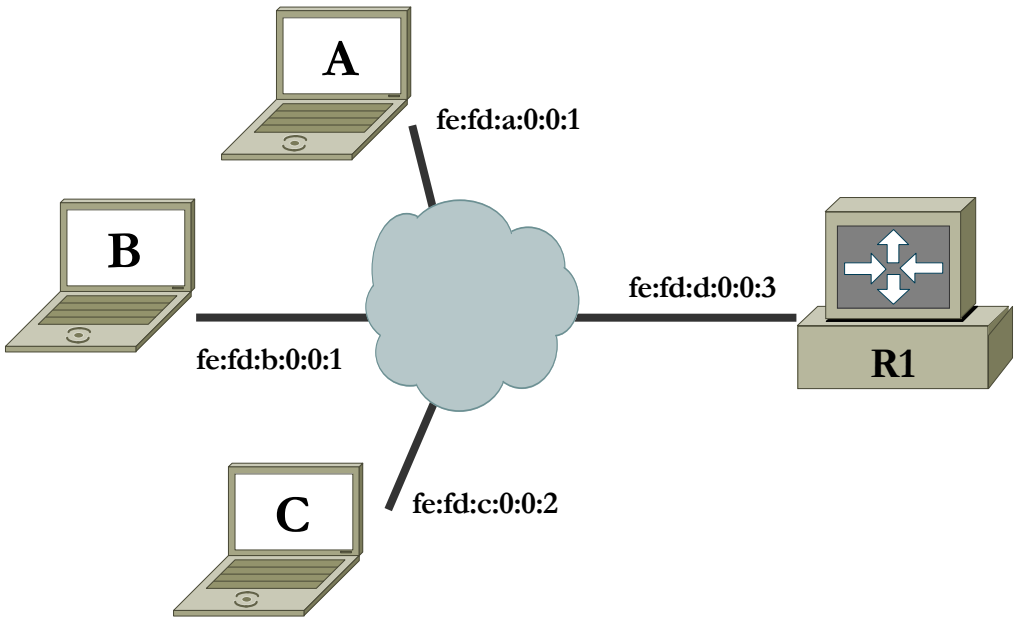

Abbildung 6.1 Szenario 1 mit Mac-Adressen

6.2.1 Erfolgsfall

Zunächst betrachten wir den Fall, dass die automatische Konfiguration erfolgreich verläuft. Da diese jedoch automatisch und schon beim Starten des Szenarios erfolgt, werden wir die entsprechende Schnittstelle zunächst herunterfahren, dann die Überwachung einer anderen Schnittstelle einstellen und schließlich die betreffende Schnittstelle wieder hochgefahren, so dass dort erneut die automatische Konfiguration durchgeführt wird. Dazu werden wir beispielhaft das Interface eth1 des Hosts A betrachten

Zunächst wechseln wir auf dessen Terminal (entweder mittels „`ssh a`“ oder mittels des entsprechenden x-term-Fensters) und fahren dort die Schnittstelle eth1 mit dem Befehl „**`ip -6 link set dev eth1 down`**“ herunter.

Wir lassen das Fenster offen und wechseln in einem anderen Fenster auf einen der anderen Hosts (z.B. B) und starten dort tcpdump auf der Schnittstelle eth1 mit dem Befehl „**`tcpdump –i eth1`**“.

Nun wechseln wir zurück ins erste Fenster und fahren das Interface wieder herauf mit dem Befehl „**`ip -6 link set dev eth1 up`**“.

Wir erhalten die folgende Ausgabe:

```
[...]
IP6   ::   >   ff02::1:ff00:1:   ICMP6,   neighbor   solicitation,   who   has
fe80::fcfd:aff:fe00:1, length 24
[...]
```

Was wir sehen ist nur die *duplicate adress detection*, doch intern ist noch einiges mehr abgelaufen. Zunächst einmal hat Host A aus seiner Mac-Adresse **fe:fd:a:0:0:1** mittels des EUI-64-Verfahrens, das in Abschnitt 3.1 erläutert wird, die linklokale Adresse **fe80::fcfd:aff:fe00:1** generiert. Für diese Adresse musste also als nächstes die *duplicate adress detection* durchgeführt werden. Als nächsten Schritt hat er dann die *Solicided-Nodes-Multicast*-Adresse, wie in Abschnitt 3.2 beschrieben, erstellt. Diese lautet **ff02::1:ff00:1**. Dieser Multicastgruppe ist A nun zunächst beigetreten und hat dann an eben diese Multicast-Adresse eine Nachbarschaftsanfrage gesendet, in der er fragte, wer die von ihm gewünschte Adresse benutzt. In diesem Fall hat Host B zwar die Multicast-Nachricht gehört, da die Multicast-Adresse, die zu seiner linklokalen Adresse gehört, ebenfalls ff02::1:ff00:1 lautet – er antwortete aber

nicht, da er die angefragte Adresse nicht verwendet hat. Da auch niemand anders geantwortet hat, wurde damit nach einiger Zeit die Konfiguration der Schnittstelle erfolgreich abgeschlossen. Dies kann beispielsweise mithilfe des Befehls „**ifconfig eth1**“ oder „**ip -6 addr show dev eth1**“ betrachtet werden. Die Ausgabe des letzteren Befehls lautet wie folgt:

```
eth1: <BROADCAST,MULTICAST,UP,10000> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether fe:fd:0a:00:00:01 brd ff:ff:ff:ff:ff:ff
    inet6 fe80::fcfd:aff:fe00:1/64 scope link
       valid_lft forever preferred_lft forever
```

Falls die Autokonfiguration noch nicht abgeschlossen ist, also bis etwa 2 Sekunden nach dem erneuten Hochfahren der Schnittstelle, erhält man hingegen die folgende Ausgabe:

```
5: eth1: <BROADCAST,MULTICAST,UP,10000> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether fe:fd:0a:00:00:01 brd ff:ff:ff:ff:ff:ff
    inet6 fe80::fcfd:aff:fe00:1/64 scope link tentative
       valid_lft forever preferred_lft forever
```

Das Schlüsselwort **tentative** bedeutet, dass die Adresse noch nicht angenomen ist, sondern sich derzeit in der Schwebe befindet.

6.2.2 Fehlerfall

Interessanter wird der Fall, falls die *duplicate adress detection* fehlschlägt. Dies wollen wir im nächsten Beispiel provozieren. Dazu werden wir wieder die Schnittstelle von Host A herunterfahren und dann die von ihm gewünschte Adresse **fe80::fcfd:aff:fe00:1** der Schnittstelle eines anderen Rechners zuweisen. Wenn wir die Schnittstelle von Host A dann reaktivieren, wird die Überprüfung auf doppelte Adressen fehlschlagen.

Zunächst wechseln wir auf Host A und fahren dort die Schnittstelle eth1 mit dem Befehl „**`ip -6 link set dev eth1 down`**“ herunter.

Dann wechseln wir auf einen anderen Rechner (z.B. Host B) und weisen dessen Schnittstelle eth1 manuell die linklokale Adresse von Host A zu. Dies geschieht mittels des Befehls: „**`ip addr add dev eth1 fe80::fcfd:aff:fe00:1`**“. Nun sollte „**`ip -6 addr show eth1`**“ folgende Ausgabe liefern:

```
5: eth1: <BROADCAST,MULTICAST,UP,10000> mtu 1500 qlen 1000
    inet6 fe80::fcfd:aff:fe00:1/128 scope link
       valid_lft forever preferred_lft forever
    inet6 fe80::fcfd:bff:fe00:1/64 scope link
       valid_lft forever preferred_lft forever
```

Ebenfalls auf Host B starten wir nun tcpdump („**tcpdump –i eth1**“)

Nun wechseln wir wieder das Fenster und fahren auf Host A die Schnittstelle wieder hinauf („**ip -6 link set dev eth1 up**“)

Das Ergebnis des tcpdump lautet wie folgt:

```
[...]
IP6   ::   >   ff02::1:ff00:1:   ICMP6,   neighbor   solicitation,   who   has
fe80::fcfd:aff:fe00:1, length 24

IP6 fe80::fcfd:aff:fe00:1 > ip6-allnodes: ICMP6, neighbor advertisement, tgt
is fe80::fcfd:aff:fe00:1, length 32
[...]
```

Die Nachbarschaftsanfrage sieht genauso aus wie im obigen Beispiel, allerdings antwortet Host B auf diese Anfrage, weil er selbst die gewünschte Adresse ja verwendet. Er antwortet also mit einer Multicast-Nachricht an alle Knoten auf dem Link, dass er die angefragte Adresse verwendet. Host A kann seine automatische Konfiguration also nicht erfolgreich abschließen, dies lässt sich dort mithilfe des Befehls „**ip -6 addr show dev“** auch schön sehen. Die Ausgabe lautet wie folgt:

```
5: eth1: <BROADCAST,MULTICAST,UP,10000> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether fe:fd:0a:00:00:01 brd ff:ff:ff:ff:ff:ff
    inet6 fe80::fcfd:aff:fe00:1/64 scope link tentative
       valid_lft forever preferred_lft forever
```

Die entsprechende Adresse ist also immer noch schwebend.

Achtung: Sowohl „ifconfig“ als auch „ip link“ zeigen nicht, dass die Konfiguration gescheitert ist, dies könnte zu unerwarteten Fehlern führen.

Um den Konflikt zu lösen und mit dem nächsten Beispiel fortfahren zu können, besteht zum einen die Möglichkeit das Szenario zu beenden und neu zu starten. Die andere Möglichkeit ist es, auf dem Host B die Adresse fe80::fcfd:aff:fe00:1/64 (mit dem Befehl: „**ip addr del dev eth1**

`fe80::fcfd:aff:fe00:1`“) zu entfernen und daraufhin die Schnittstelle von Host A herunter- und noch einmal herauf zufahren (wie in den beiden vorhergehenden Beispielen)

6.3 Zustandslose Autokonfiguration globaler Adressen

Im folgenden Beispiel werden wir die Autokonfiguration globaler Adressen mittels Routerankündigungsnachrichten untersuchen. Dazu wird jedoch ein *Router Advertising Demon* benötigt, der zunächst installiert werden muss.

6.3.1 Installation

Zur Installation bieten sich verschiedene Möglichkeiten an, je nach verwendeter Linux-Distribution als Gastsystem in den UML-Maschinen.

- Die Installation aus den Quellen, sollte immer funktionieren.
- Bei Debian und Ubuntu bietet sich das entsprechende .deb-Paket an.
- Unter Suse, Red Hat und Mandriva bietet sich die Installation über das entsprechende .rpm-Paket an.

Unabhängig von der gewählten Installationsmethode muss zunächst einmal der Gastrechner gestartet werden, wobei das Filesystem als „direct“ und nicht wie sonst üblich als „copy-on-write“ gemounted werden muss. Eine Beispielkonfiguration ist im Anhang in Abschnitt 8.1.2 zu finden.

Nach dem Kopieren der Installationsdaten mithilfe des „start“-Tags („**`vnumlparser.pl –x start@patchday.xml`**“) folgt man den entsprechenden Anweisungen und löscht danach die Installationsdateien wieder („**`vnumlparser.pl –x stop@patchday.xml`**“)

6.3.1.1 Installation aus den Quellen

Die Quelldateien können unter http://www.litech.org/radvd/ herunter geladen werden. Am besten entpackt man sie noch auf dem Hostrechner. Nach dem Kopieren auf das Gastsystem wechselt man dort in das entsprechende Verzeichnis („**`cd /install/`**“) und startet dann nacheinander

1. `./configure`
2. `./make`
3. `./make install`

6.3.1.2 Installation aus einem .deb-Paket

Das benötigte .deb-Paket kann mittels apt-get heruntergeladen werden. Das benötigte Paket heißt radvd. Der Befehl für den Download lautet also „**`apt-get install radvd -d`**“ und sollte auf dem Hostrechner ausgeführt werden, da das Gastsystem in der Regel keine Verbindung zum Internet hat. Das Debian-Paket liegt dann im Verzeichnis **`/var/cache/apt/archives`** und muss von dort auf das Gastsystem kopiert werden. Dort kann man es dann mittels „**`dpkg –i radvd_1%3a0.8-1ubuntu2_i386.deb`**“ installiert werden

6.3.1.3 Installation aus einem .rpm-Paket

Das .rpm-Paket kann von http://www.litech.org/radvd/ herunter geladen werden. Nach dem kopieren aufs Gastsystem installiert man es mit „**`rpm –i radvd-1.0-1.rhl9.i386.rpm`**“

6.3.2 Konfiguration

Die Konfigurationsdatei des *router advertisement daemon* (kurz: **Radvd)** besteht aus einem oder mehreren Schnittstellen-Definitionen. Die Schnittstellenbezogenen Optionen bestimmen unter anderem, ob und wie oft Routerankündigungen gesendet werden und ob DHCP für die zustandsbehaftete Adresskonfiguration oder/und für die Konfiguration von weiteren Parametern (wie etwa die Adresse von DNS-Servern) verwendet werden soll.

```
interface <Name>
{
      <Schnittstellen-bezogene Optionen>
      <beliebig viele Präfix-Definitionen>
      <beliebig viele Routen-Definitionen>
};
```

Präfix-Definitionen bestimmen welche Präfixe auf dem Link gültig sind und in den Optionen wird festgelegt, wie deren Lebenszeiten dimensioniert sind. Sie haben die Form:

```
prefix <Präfix>/<Länge>
{
```

```
      <Präfix-bezogene Optionen>
};
```

Standardmäßig tragen Hosts einen der Router, von dem sie eine Router-Ankündigung erhalten als Default-Router ein, an den sie Pakete schicken, an die Netze zu deren Zieladressen sie nicht direkt verbunden sind. Allerdings ist es möglich auch detailliertere Routinginformationen zu übertragen. Dies geschieht mit den Routen-Definitionen, die die folgende Form haben:

```
route Präfix/Länge
{
      <Routen-bezogene Optionen>
};
```

Die wichtigsten Optionen im Überblick, weitere Optionen sind auf der man-Page zu radvd.conf nachzulesen.

Schnittstellen-bezogenen Optionen

AdvSendAdvert on|off; Wenn diese Option auf den Wert on gesetzt ist, werden Routerankündigungen periodisch sowie als Antwort auf Routeranfragen gesendet (Standard: off)

AdvManagedFlag on|off; Wenn diese Option auf den Wert on gesetzt ist, wird DHCPv6 verwendet, um zusätzlich zu den zustandslos erzeugten Adressen weitere zu erstellen. (Standard: off)

AdvOtherConfigFlag on|off; Wenn diese Option auf den Wert on gesetzt ist, wird DHCPv6 verwendet, um zusätzliche Konfigurationsparameter zu übertragen. (Standard off)

Präfix-bezogene Optionen:

AdvValidLifetime <Sekunden>|infinity; Zeit, die eine mithilfe dieses Präfixes erstellte Adresse gültig bleibt. Standard ist 30 Tage.

AdvPreferredLifetime <Sekunden>|infinity; Zeit, die eine mithilfe dieses Präfixes erstellte Adresse bevorzugt bleibt. Standard ist 7 Tage.

Routen-bezogene Optionen:

AdvRoutePreference low|medium|high; Präferenz mit der dieser Router als Next-Hop für das Präfix gewählt wird, wenn mehrere Router dieses Präfix anbieten. Standard ist medium.

Eine sehr einfache Konfigurationsdatei ist in folgendem Beispiel gegeben. Routerankündigungen werden aktiviert und das Präfix 3ffe::/64 dem Link zugeteilt.

```
interface eth1
{
      AdvSendAdvert on;
      prefix 3ffe::/64
      {
      };
};
```

Beispiel 6.1

Achtung: Es ist darauf zuachten, dass die Zeilenumbrüche im Unix-Stil erfolgen.

6.3.3 Starten des Dienstes

Radvd wird mithilfe des folgenden Befehls gestartet: „**/usr/sbin/radvd –C <Konfigurationsdatei>**". Bevor wir ihn allerdings wirklich ausführen, starten wir noch auf einem der Hosts (zum Beispiel B) tcpdump auf der Schnittstelle eth1. Wir betrachten nun zunächst den Fall, dass der Dämon das erste Mal gestartet wird. Darauf werden wir noch den Fall untersuchen, wenn eine Schnittstelle neu hochgefahren wird, da die Folge der Nachrichten dann abweicht.

6.3.3.1 Router wird zuerst hochgefahren

Nachdem wir den Daemon in unserem Netz gestartet haben, erhalten wir die folgende Ausgabe des tcpdump.

```
IP6 fe80::fcfd:dff:fe00:3 > ip6-allnodes: ICMP6, router advertisement, length
56
...
```

```
IP6 :: > ff02::1:ff00:2: ICMP6, neighbor solicitation, who has
3ffe::fcfd:cff:fe00:2, length 24

IP6 :: > ff02::1:ff00:1: ICMP6, neighbor solicitation, who has
3ffe::fcfd:aff:fe00:1, length 24

IP6 :: > ff02::1:ff00:1: ICMP6, neighbor solicitation, who has
3ffe::fcfd:bff:fe00:1, length 24
...
```

Was wir sehen ist zunächst einmal die erste periodische (unangeforderte) Routerankündigung. Diese wird an alle Knoten auf dem Link gesendet. Alle 3 Hostcomputer haben noch keine automatisch konfigurierte Adresse mit dem gegeben Präfix. Deshalb bilden sie die entsprechenden Adressen und führen daraufhin eine DAD durch. Diese Überprüfungen sind in den nächsten 3 Zeilen zu sehen. Niemand antwortet darauf, also werden die Adressen kurze Zeit später angenommen. Zur Überprüfung dient wieder der Befehl: „**`ip addr show dev eth1`**“. Er liefert beispielsweise auf Host A die folgende Ausgabe:

```
5: eth1: <BROADCAST,MULTICAST,UP,10000> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether fe:fd:0a:00:00:01 brd ff:ff:ff:ff:ff:ff
    inet6 3ffe::fcfd:aff:fe00:1/64 scope global dynamic
       valid_lft 2591822sec preferred_lft 604622sec
    inet6 fe80::fcfd:aff:fe00:1/64 scope link
       valid_lft forever preferred_lft forever
```

Auch die default-Route wird wie erwartet eingetragen, wie der Aufruf von „**`ip -6 route show`**“ zeigt:

```
...
default via fe80::fcfd:dff:fe00:3 dev eth1  proto kernel  metric 1024  expires
1781sec mtu 1500 advmss 1440 hoplimit 64
```

6.3.3.2 Schnittstelle wird zuerst hochgefahren

Nun betrachten wir den Fall, dass eine Schnittstelle (etwa eth1 von Host A) hochgefahren wird, während der Router bereits aktiv ist. Dazu fahren wir sie zunächst herunter („**`ip link set dev eth1 down`**“) und dann wieder hoch („`ip link set dev eth1 up`“). Zu beachten ist hierbei, dass der tcpdump auf Host B noch läuft. Dessen Ausgabe sieht dann wie folgt aus:

```
...
IP6 :: > ff02::1:ff00:1: ICMP6, neighbor solicitation, who has
fe80::fcfd:aff:fe00:1, length 24
...

IP6 fe80::fcfd:aff:fe00:1 > ip6-allrouters: ICMP6, router solicitation, length
16

IP6 fe80::fcfd:dff:fe00:3 > ip6-allnodes: ICMP6, router advertisement, length
56

IP6 :: > ff02::1:ff00:1: ICMP6, neighbor solicitation, who has
3ffe::fcfd:aff:fe00:1, length 24
```

Zunächst führt Host A eine DAD für seine linklokale Adresse aus. Parallel dazu wird auch schon eine Routeranforderung gesendet, um etwaige Verzögerungen durch eine verspätete Routerankündigung möglichst zu vermeiden. Daraufhin sendet der Router eine Routerankündigung an alle Knoten auf dem Link, die A auch dann empfangen könnte, wenn er die DAD für seine linklokale Adresse noch nicht abgeschlossen hätte. Nachdem nun die linklokale Adresse angenommen wurde, kann der Knoten die globale Adresse bilden und erneut eine DAD durchführen. Da wieder niemand antwortet, sind beide Adressen nun gültig.

6.4 DHCPv6

Als nächstes werden wir uns mit dem DHCPv6-Dienst beschäftigen. Die Auswahl an Implementationen ist immer noch sehr eingeschränkt. Die einzigen beiden freien Implementationen für Linux sind:

- Wide-DHCPv6[3] und
- Dibbler[4]

Im Rahmen dieser Arbeit wird die Benutzung von Dibbler erläutert, da es einfacher zu konfigurieren ist sowie zum Zeitpunkt der Erstellung dieser Arbeit besser gepflegt wurde.

[3] http://sourceforge.net/projects/wide-dhcpv6/
[4] http://sourceforge.net/projects/dibbler/

6.4.1 Installation

Die Quelldateien können auf der Sourceforge-Seite herunter geladen werden. Zunächst müssen die Daten wie in Abschnitt 6.3.1 auf das Gastsystem kopiert werden. Danach wechselt man in das entsprechende Verzeichnis und führt nacheinander die folgenden Befehle aus:

- make client server relay doc
- make install
- mkdir -p /var/lib/dibbler

Alternativ gibt es auch die Debian-Pakete: dibbler-client, dibbler-server, dibbler-relay und dibbler-doc

Nach der Installation stehen die folgenden Manpages zur Verfügung: dibbler-client, dibbler-server und dibbler-relay.

6.4.2 Konfiguration

Im Gegensatz zur zustandslosen Autokonfiguration nur mit Routerankündigungen, müssen neben dem Router auch die Host-Rechner konfiguriert werden. Auf dem Router wird in der Regel ein DHCPv6-Server laufen, auf den Hosts DHCPv6-Clients. Die Konfigurationsdateien sollen im Ordner /etc/dibbler/ liegen.

6.4.2.1 Syntax

Im Folgenden werden die einzelnen Konfigurationsdateien beschrieben. Dabei werden einige Datentypen benutzt, die wie folgt definiert sind:

`DUID` ist eine Hexadezimale Zahl, die mit 0x beginnt.

Beispiel: 0xabcd11

`DUID-Liste` sind eine oder mehrere DUIDs mit Kommata getrennt.

Beispiel: 0xabcd11, 0xaffe

`Adress-Liste` ist eine Liste von IPv6-Adressen mit Kommata getrennt.

Beispiel: 3ffe::1, 2001::2

Außerdem bedeutet der senkrechte Strich „|“ eine Alternative.

Beispiel: **`rapid-commit <YES | NO>`** bedeutet, dass mögliche Ausprägungen dieser Option **`rapid-commit YES`** und **`rapid-commit NO`** sind.

6.4.2.2 Client

Die Konfigurationsdatei eines DHCPv6-Clients heißt **`/etc/dibbler/client.conf`**. Es besteht eine sehr einfache Möglichkeit der Konfiguration, nämlich eine leere Konfigurationsdatei. In diesem Fall wird versucht, für alle Schnittstellen jeweils eine Adresse zu beziehen. Ansonsten enthält die Datei neben globalen Optionen einen Schnittstellen-Deklarations-Block für jede Schnittstelle, für die eine Adresse bezogen werden soll. Die Datei ist also wie folgt aufgebaut:

```
<globale Optionen>
<Schnittstellen-Optionen>
<IA-Optionen>
<Adress-Optionen>
<eine oder mehrere Schnittstellen-Deklarationen>
```

Wenn an dieser Stelle Schnittstellen-, IA- und Adress-Optionen deklariert werden, dann gelten sie als Standardwerte für die jeweiligen Blöcke.

Es existiert eine Schnittstellen-Deklaration für jede Schnittstelle, für die Adressen oder/und Konfigurationsparameter bezogen werden sollen. Ein solcher Block wie folgt aufgebaut:

```
iface <Schnittstellen-Name>
{
      <Schnittstellen-Optionen>
      <IA-Optionen>
      <Adress-Optionen>
      <beliebig viele IA oder TA-Deklarationen>
}
```

Eine IA bzw. TA-Deklaration wiederum besteht aus beliebig viele Adress-Deklarationen. Wenn das Schlüsselwort „ia“ verwendet wird, werden normale (nicht temporäre) Adressen angefordert. Wird hingegen das Schlüsselwort „ta“ verwendet, werden temporäre Adressen angefordert. werden Sowohl ia- als auch ta-Blöcke sind wie folgt aufgebaut:

```
<ia|ta> <Anzahl>
{
      <IA-Optionen>
      <Adress-Optionen>
      <beliebig viele Adress-Deklarationen>
}
```

<Anzahl> bestimmt die Anzahl der IAs, mit diesen Parametern, die angefordert werden sollen. Wenn an dieser Stelle nichts angegeben wird, wird angenommen, dass eine IA angefordert werden soll. Sollte <Anzahl> jedoch größer als 1 sein, dann sind Adress-Deklarationen nicht erlaubt.

Adress-Deklaration:

```
adress <Anzahl>
{
      <Adress-Optionen>
      <IPv6-Adresse>
}
```

<Anzahl> bestimmt die Anzahl der Adressen mit diesen Parametern, die angefordert werden sollen. Falls nichts angegeben wurde, wird 1 angenommen. Wenn <Anzahl> größer ist als 1 darf keine <IPv6-Adresse> angegeben werden. Eine solche Adresse dient als Hinweis für den Server, dass der Client gerne diese Adresse hätte.

Die wichtigsten Optionen im Überblick:

Globale Optionen

- **`stateless`** : Wenn diese Option definiert ist, dann wird der Client keine Adresse über DHCPv6 beziehen. IA-Deklarationen sind dann also nicht erlaubt. Wenn sie nicht definiert

wird und außerdem keine IA-Deklaration vorhanden, geht Dibbler laut Spezifikation davon aus, dass eine Adresse je Schnittstelle bezogen werden soll. Dieses Verhalten konnte jedoch in Tests nicht bestätigt werden, demnach wird bei fehlender IA-Deklaration keine Adresse bezogen.

- **log-level <1 .. 8>** : Dieser Wert bestimmt, wie viele Protokollnachrichten aufgezeichnet werden. Wobei größere Zahlen mehr Protokollnachrichten bedeuten. Standard ist 7.

- **duid-type <DUID-LLT | DUID-EN <Enterprise-Nummer> <Enterprise ID>| DUID – LL>** : Dieser Parameter bestimmt, welche Art von DUID generiert werden soll, wenn noch keine DUID existiert. Standard ist DUID-LLT.

Schnittstellen-Optionen

- **rapid-commit <YES | NO>** :Dieser Wert bestimmt, ob 2-Wege-Kommunikation erlaubt ist. Auch der Server muss diese Option aktivieren. Standard ist NO.

- **unicast <YES | NO>** : Diese Option bestimmt, ob der Client Unicast-Kommunikation mit dem Server anfordern soll. Falls der Server dies unterstützt wird er dies in seiner Antwort mitteilen. Standard ist NO.

- **prefered-servers <DUID-Liste | Adress-Liste>** : Diese Option bestimmt, welche Server bevorzugt werden sollen. Falls mehrere Server auf eine SOLICIT-Nachricht antworten, wird er diejenigen auf dieser Liste bevorzugen.

- **reject-servers <DUID-LISTE | Adress-Liste>** : Diese Option bestimmt, welche Server ignoriert werden müssen.

- **option dns-server** : wenn diese Option angegeben wurde, wird der Client versuchen, eine DNS-Adresse vom Server zu beziehen.

Weitere Optionen sind im Dibbler-User's Guide [DIBBLER] zu finden.

Beispielkonfigurationen:

Das folgende Beispiel zeigt eine Konfigurationsdatei, die sowohl eine eigene Adresse als auch die der verfügbaren DNS-Server anfordert.

```
# client.conf
iface eth1
{
      ia
      option dns-server
}
```

Beispiel 6.2

Das folgende Beispiel zeigt eine Konfigurationsdatei, die keine Adresse anfordert sondern nur die Adresse der vorhandenen DNS-Server beschafft.

```
# client.conf
stateless
iface eth1
{
      option dns-server
}
```

Beispiel 6.3

Das letzte Beispiel zeigt eine Konfigurationsdatei, die 2 Adressen anfordert. Die erste Adresse soll möglichst 3ffe::1:1 lauten, die zweite Adresse ist nicht weiter spezifiziert. Außerdem wird die Adresse der verfügbaren DNS-Server angefordert.

```
# client.conf
iface eth1
{
      ia
      {
            adress
            {
                  3ffe::1:1
            }
            adress
      }
      option dns-server
}
```

Beispiel 6.4

6.4.2.3 Server

Die Konfigurationsdatei eines DHCPv6-Servers heißt **`/etc/dibbler/server.conf`**. Sie enthält – neben globalen Optionen – einen Abschnitt, für jede Schnittstelle, die verwaltet werden soll.

```
<globale Optionen>
<Schnittstellen-Optionen>
<Klassen-Optionen>
<eine oder mehrere Schnittstellen-Deklarationen>
```

Wenn Schnittstellen- und Klassen-Optionen definiert werden, dann dienen sie als Standardwerte für alle Schnittstellen bzw. Klassen dieses Servers.

Schnittstellen-Deklarationen wiederum enthalten wiederum Optionen und Klassendeklarationen. Klassen sind Gruppen von Adressen, die jeweils einzeln Optionen enthalten können. Es ist zu beachten, dass Relay-Schnittstellen genauso konfiguriert werden.

```
iface <name>
{
      <Schnittstellen-Optionen>
      <Klassen-Optionen>
      <eine oder mehrere Klassen-Deklarationen>
}
```

Klassen-Deklarationen sind wie folgt aufgebaut:

```
class
{
<Klassen-Optionen>
<ein oder mehrere Adress-Pool-Deklarationen>
}
```

Eine Adress-Pool Deklaration hat die Form:

```
pool <minaddress> - <maxaddress>
```

oder

```
pool <Präfix>/<Länge>
```

Dabei kann die Reihenfolge der Zeilen innerhalb der einzelnen Blöcke vertauscht werden.

Die wichtigsten Optionen im Überblick:

Globale Optionen

- **`log-level <1 .. 8>`** : Diese Option bestimmt, wie viele Protokollnachrichten aufgezeichnet werden sollen.. Größere Zahlen stehen dabei für mehr Nachrichten. Standard ist 7.

- **`stateless`** : Diese Option bestimmt, ob der Server lediglich Konfigurationsdaten zur Verfügung stellen oder auch Adressen vergeben soll. Falls diese Option nicht gesetzt ist, muss zu jeder Schnittstelle mindestens eine Klassendeklaration vorhanden sein.

Schnittstellen-Optionen

- **`preference <0 .. 255>`**: <Wert> ist eine Zahl zwischen 0 und 255. Wenn ein Client mehrere ADVERTISE-Nachrichten erhält, soll er denjenigen Server mit dem höchsten Wert auswählen. Ausserdem soll ein Client nicht warten, ob noch andere ADVERTISE-Nachrichten eingehen, sobald eine Nachricht mit einem preference-Wert von 255 eingeht. Standardmäßig ist dies deaktiviert.

- **`unicast <Adresse>`** : Normalerweise senden Clients Daten an die „Alle DHCP-Relay-Agenten und Server"-Multicast-Adresse. Dies führt allerdings dazu, dass alle Knoten diese Nachrichten verarbeiten müssen. <Adresse> ist die IPv6-Adresse des DHCP-Servers, so dass Clients, die dies Unterstützen ihre Daten per Unicast an ihn senden können. Standardmäßig ist dies deaktiviert.

- **`rapid-commit <YES|NO>`**: Wenn diese Option den Wert YES hat, dann wird 2-Wege-Kommunikation eingesetzt, wenn auch der Client 2-Wege-Kommunikation unterstützt. Sonst wird 4-Wege-Kommunikation eingesetzt.

- **`iface-max-lease <Wert>`** : Dieser Wert legt fest, wie viele Adressen für diese Schnittstelle insgesamt vergeben werden sollen Standard ist 2^{32}-1.

- **client-max-lease <Wert>** : Dieser Wert legt fest, wie viele Adressen an einen einzelnen Client vergeben werden sollen. Dies soll hauptsächlich vor falsch konfigurierten Clients schützen. Standard ist 2^{32}-1.

- **relay <Schnittstellen-Name>** : Wird für Relay-Definitionen verwendet. Diese Optionen bezeichnet die physikalische Schnittstelle bzw. eine andere Relay-Schnittstelle (bei kaskadierenden Relays) über die Daten für diese Relay-Schnittstelle empfangen und versendet werden. Standardmäßig ist dies nicht definiert.

- **interface-id <Wert>** : Wird für Relay-Definitionen verwendet. Jede Relay-Schnittstelle benötigt eine eindeutige Nummer. Standardmäßig ist dies nicht definiert.

- **option dns-server <Adressen-Liste>** : Diese Option liefert Informationen über verfügbare DNS-Server.

- **option ntp-server <Adressen-Liste>** : Diese Option liefert Informationen über verfügbare NTP-Server.

Klassen-Optionen

- **T1 <Wert>** oder **T1 <MinWert> - <MaxWert>** **:** Dieser Wert bestimmt, nach welcher Zeit in Sekunden ein Client einen Renew-Prozess durchführen soll. Standard ist 2^{32}-1. Nicht verfügbar für temporäre Adressen.

- **T2 <Wert>** oder **T2 <MinWert> - <MaxWert>** **:** Dieser Wert bestimmt, nach welcher Zeit in Sekunden ein Client einen Rebind-Prozess durchführen soll, falls der Renew-Prozess scheitert. Standard ist 2^{32}-1. Nicht verfügbar für temporäre Adressen.

- **valid-lifetime <Wert>** oder **valid-lifetime <MinWert> - <MaxWert>** **:** Dieser Wert bestimmt die Zeit in Sekunden, die eine Adresse aus dieser Klasse gültig bleibt. Standard ist 2^{32}-1.

- **prefered-lifetime <Wert>** oder **prefered-lifetime <MinWert> - <MaxWert>** : Dieser Wert bestimmt die Zeit in Sekunden, die eine Adresse aus dieser Klasse bevorzugt bleibt. Standard ist 2^{32}-1.

- **class-max-lease <Wert>** : Dieser Wert bestimmt die Anzahl der Adressen dieser Klasse, die maximal vergeben werden sollen. Standard ist 2^{32}-1.

- **reject-clients <DUID-Liste> oder reject-clients <Liste von linklokale Adressen>** : Diese Option wird auch als Black-List bezeichnet. Der Wert bezeichnet einen Client, der nicht unterstützt werden soll. Der Client kann über seine DUID oder über seine linklokale Adresse identifiziert werden.

- **accept-only <DUID-Liste> oder accept-only <Liste von linklokale Adresse>** : Diese Option wird auch als White-List bezeichnet. Wenn dies nicht angegeben wird, werden alle Clients unterstützt, außer jenen auf der Black-List. Wenn er angegeben wird, werden nur diese Clients unterstützt.

- **share <Wert> :** Wenn mehrere Klassen auf einer Schnittstelle definiert werden, wird eine der passenden Klassen zufällig bei der Adressvergabe ausgewählt. Normalerweise haben alle Klassen die gleiche Wahrscheinlichkeit. Dies kann mit diesem Parameter beeinflusst werden. Angenommen es gibt 2 Klassen mit den Werten 100 und 300, dann werden etwa 100/(100+300)= ¼ der Adressen aus dieser Klasse gewählt. Aus der zweiten Klasse werden 300/(100+300) = ¾ der Adressen ausgewählt.

Beispielkonfigurationen:

Dieses einfache Beispiel bietet auf dem Interface eth1 35 Adressen an.

```
# server.conf
iface eth1
{
      class
      {
            pool 3ffe::1:1-3ffe::1:35
```

```
      }
}
```

Beispiel 6.5

Das nächste Beispiel zeigt die Verwendung der verschiedenen Zeitangaben. Außerdem wird ein DNS-Server angegeben.

```
# server.conf
iface eth1
{
      T1 600
      T2 900
      prefered-lifetime 2400
      valid-lifetime 4800
      class
      {
            pool 3ffe::2:0/112
      }
      option dns-server 2001::13
}
```

Beispiel 6.6

6.4.2.4 Relay-Agenten

Die Konfigurationsdatei eines Relay-Agenten liegt in /etc/dibbler/relay.conf. Eine solche Datei besteht neben globalen Optionen aus einer oder mehreren Schnittstellendeklarationen. Dabei ist zu beachten, dass ein Relay-Agent erst mit mindestens 2 Schnittstellen Sinn macht, da er sonst eingehende Pakete wieder über die gleiche Schnittstelle versenden würde. Sie ist also wie folgt aufgebaut

```
<globale Optionen>
<Schnittstellen-Optionen>
<eine oder mehrere Schnittstellendeklarationen>
```

Eine Schnittstellendeklaration hat die Form:

```
iface <Schnittstellen-Name>
{
      <Schnittstellen-Optionen>
}
```

Die wichtigsten Optionen im Überblick:

Globale Optionen

- **log-level <1 .. 8>** : Diese Option bestimmt, wie viele Protokollnachrichten aufgezeichnet werden sollen.. Größere Zahlen stehen dabei für mehr Nachrichten. Standard ist 7.

Schnittstellen-Optionen

- **client multicast <YES | NO>** : Diese Option bestimmt, ob der Relay-Agent auf dieser Schnittstelle auf die Alle-DHCP-Server-Multicast-Adresse hören soll. Standard: NO.

- **client unicast <Adresse>** : Diese Option bestimmt, ob der Relay-Agent auf eine bestimmten Unicast-Adresse auf Nachrichten lauschen soll. Dies wird hauptsächlich verwendet um zwei Relay-Agenten miteinander zu verbinden. Standard: nicht definiert.

- **server multicast <YES | NO>** : Diese Option bestimmt, ob der Relay-Agent, Nachrichten, die er von einer beliebigen Schnittstelle erhalten hat an die Multicast-Adresse ff05::1:3 senden soll. Es ist zu beachten, dass dies nicht die Adresse ist, auf die Server normalerweise lauschen (ff02::1:2)

- **server unicast <Adresse**> : Diese Option bestimmt, an welche Unicast-Adresse der Relay-Agent Nachrichten weiterleiten soll. Es ist zu beachten, dass der Server entsprechend konfiguriert werden muss, um Unicast-Nachrichten zu empfangen.

- **interface-id <Wert>** : Die Schnittstellen auf der Client-Seite benötigen jeweils eine einen eindeutigen Schnittstellenbezeichner. Es ist wichtig, dass diese mit den Bezeichnern des Servers konsistent sind.

Beispielkonfigurationen:

Das folgende einfache Beispiel lauscht auf Multicast-Nachrichten auf der Schnittstelle eth2 und sendet sie als Multicast-Nachricht auf der Schnittstelle eth1 weiter.

```
# relay.conf
iface eth1
{
      server multicast yes
```

```
}

iface eth2
{
      interface-id 4711
      client multicast yes
}
```

Beispiel 6.7

6.4.2.5 Präfixdelegation

Präfixdelegation (siehe 4.10) erfordert einige wenige Änderungen an den Konfigurationsdateien:

Der Client muss innerhalb eines Schnittstellen-Blocks ein oder mehrere Präfixe mit einem pd-Block anfordern. Ein solcher ist ähnlich aufgebaut, wie ein ia- oder ta-Block, enthält aber keinen address-Block. Im folgenden Beispiel wird beispielsweise ein Präfix angefordert.

```
# client.conf
iface eth1
{
      pd
      {
      T1 1000
      T2 2000
      }
}
```

Beispiel 6.8

Der Server muss innerhalb eines Schnittstellen-Blocks eine oder mehrere pd-class-Blöcke definieren. Diese ist fast aufgebaut wie ein normaler class-Block, enthält aber anstelle von pool-Blöcken pd-pool-Blöcke. Darüber hinaus muss auch die Länge der vergebenen Präfixe mithilfe einer pd-length-Option angegeben werden. Im folgenden Beispiel wird für jeden Client ein Präfix der Länge 64-Bit aus dem Bereich 4711::/48 vergeben.

```
# server.conf
iface eth1
{
      pd-class
      {
```

```
        pd-pool 4711::/48
        pd-length 64

        }
}
```

Beispiel 6.9

6.4.3 Anwendungsbeispiele

Im Folgenden werden einige einfache Szenarien vorgestellt, mit denen die Kommunikation zwischen den verschiedenen DHCP-Komponenten untersucht werden kann.

6.4.3.1 4-Wege-Kommunikation

Zunächst einmal wollen wir die Kommunikation zwischen einem Client und einem Server untersuchen, die die 4-Wege-Kommunikation verwenden, dazu werden wir ein einfaches 2-Knoten-Szenario starten und dort jeweils einen Server und einen Client mit einer sehr einfachen Konfiguration ausführen. Die folgende Abbildung zeigt das verwendete Szenario:

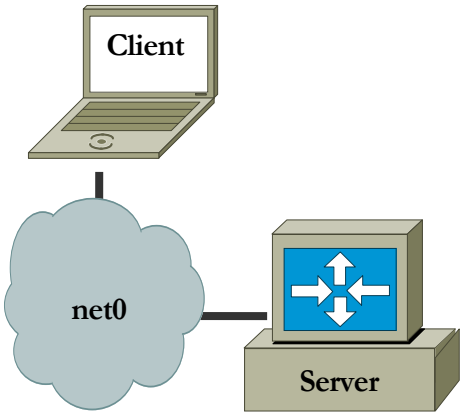

Abbildung 6.2 Aufbau des ersten DHCP-Beispiel-Szenarios

Zunächst starten wir also das Szenario aus Anhang 8.1.3. Dabei verwenden wir, die Dibbler-Konfigurationsdateien aus Anhang 8.2.1.

Zur Überwachung starten wir zunächst „tcpdump –i eth1“ auf „Client“ (oder auf „Server“), danach starten wir auf „Server“ mithilfe des Befehles „dibbler-server run“ den DHCP-Server. Die Ausführung

mit dem Kommando „run“ führt den Server hierbei als Konsolenanwendung aus. Als Dämon (also im Hintergrund) würde man in mit „dibbler-server start“ ausführen. Da wir den Komponenten jedoch bei der Arbeit zuschauen möchten, ist die Ausführung als Konsolenanwendung sinnvoll.

Danach starten wir den DHCPv6-Client auf „Client“, was analog zum Starten des Servers verläuft. Allerdings heißt die auszuführende Datei nahe liegender Weise „dibbler-client“.

Der tcpdump liefert das folgende Ergebnis:

```
[…]
IP6 fe80::fcfd:ff:fe00:101.546 > ff02::1:2.547: dhcp6 solicit
IP6 fe80::fcfd:ff:fe00:201.547 > fe80::fcfd:ff:fe00:101.546: dhcp6 advertise
IP6 fe80::fcfd:ff:fe00:101.546 > ff02::1:2.547: dhcp6 request
IP6 fe80::fcfd:ff:fe00:201.547 > fe80::fcfd:ff:fe00:101.546: dhcp6 reply
[…]
IP6 :: > ff02::1:ff00:1ba: ICMP6, neighbor solicitation, who has 4711::1ba,
length 24
[…]
```

Wie zu sehen ist, wurde die 4-Wege-Kommunikation (bestehend aus Solicit, Avertise, Request und Reply) ausgeführt. Danach wurde noch eine DAD durchgeführt und die Adresse daraufhin angenommen.

Außerdem wurden zwei DNS-Server eingetragen. Dies kann man in der Datei /etc/resolv.conf nachsehen. Diese sollte nun die folgenden Zeilen beiden Zeilen enthalten:

```
nameserver 2000::100
nameserver 2000::101
```

6.4.3.2 2-Wege-Kommunikation

In diesem Beispiel soll nun die 2-Wege-Kommunikation zum Einsatz kommen, dabei kann das gleiche Szenario verwendet werden, allerdings mit leicht modifizierten Konfigurationsdateien für Client und Server (Anhang 8.2.2)

Der analog zum obigen Beispiel ausgeführte tcp-dump liefert:

```
IP6 fe80::fcfd:ff:fe00:101.546 > ff02::1:2.547: dhcp6 solicit
IP6 fe80::fcfd:ff:fe00:201.547 > fe80::fcfd:ff:fe00:101.546: dhcp6 reply
[…]
IP6 :: > ff02::1:ff00:1bb: ICMP6, neighbor solicitation, who has 4711::1bb,
length 24
[…]
```

Es wird also wie erwartet eine 2-Wege-Kommunikation (Solicit, Reply) ausgeführt und danach wieder eine DAD.

6.4.3.3 Zustandsloses DHCP

Das folgende Beispiel soll zeigen, wie Client und Server so eingestellt werden können, damit nur Konfigurationsdaten nicht aber Adressen konfiguriert werden. Die Serverkonfiguration müsste dazu im vergleich zu den beiden vorhergehenden Beispielen nicht unbedingt verändert werden, allerdings kann sie nun kürzer ausfallen.

Wir verwenden wieder dasselbe Szenario wie in den beiden vorhergegangen Beispielen und benutzen nun die Konfigurationsdateien aus Anhang 8.2.3.

Tcp-dump liefert in diesem Fall:

```
[…]
IP6 fe80::fcfd:ff:fe00:101.546 > ff02::1:2.547: dhcp6 inf-req
[…]
IP6 fe80::fcfd:ff:fe00:201.547 > fe80::fcfd:ff:fe00:101.546: dhcp6 reply
[…]
```

6.4.3.4 Relay-Agenten-Beispiel

Für das folgende Beispiel, das den Gebrauch von Relay-Agenten demonstrieren soll, verwenden wir ein geringfügig größeres Szenario. Das in der folgenden Abbildung gezeigt wird:

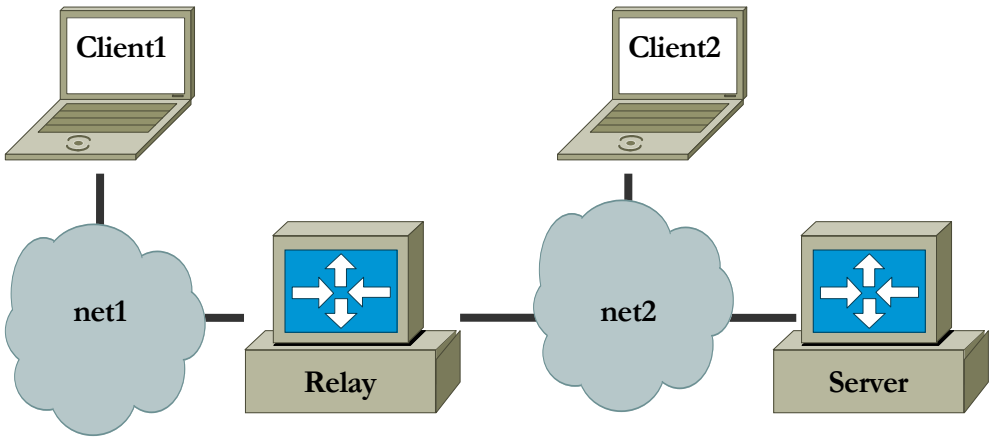

Abbildung 6.3 Aufbau des Relay-Agenten-Beispiels

Nun setzen wir zunächst die tcpdumps (zum Beispiel jeweils auf die Schnittstellen eth1 der Hosts Client1 und Server und starten danach wie gewohnt den Client, den Server und mithilfe des Befehls „dibbler-relay run“ auch den Relay-Agent..

Der erste tcpdump (in Netz net1) liefert das folgende Ergebnis:

```
IP6 fe80::fcfd:ff:fe00:101.546 > ff02::1:2.547: dhcp6 solicit
[…]
IP6 fe80::fcfd:ff:fe00:301.547 > fe80::fcfd:ff:fe00:101.546: dhcp6 advertise
IP6 fe80::fcfd:ff:fe00:101.546 > ff02::1:2.547: dhcp6 request
IP6 fe80::fcfd:ff:fe00:301.547 > fe80::fcfd:ff:fe00:101.546: dhcp6 reply
[…]
21:56:55.654773 IP6 :: > ff02::1:ff00:33d: ICMP6, neighbor solicitation, who
has 4000::33d, length 24
```

Wie zu sehen ist, sieht diese Kommunikation genauso aus, als wenn sich der Server in diesem Netz befinden würde. Für den Client ist nicht zu erkennen, dass er mit einem Relay-Agent kommuniziert.

Der tcpdump im Netz „net2“ liefert das folgende Ergebnis:

```
IP6 fe80::fcfd:ff:fe00:302.547 > ff05::1:3.547: dhcp6 relay-fwd
[…]
IP6 fe80::fcfd:ff:fe00:401.547 > fe80::fcfd:ff:fe00:302.547: dhcp6 relay-reply
IP6 fe80::fcfd:ff:fe00:302.547 > ff05::1:3.547: dhcp6 relay-fwd
IP6 fe80::fcfd:ff:fe00:401.547 > fe80::fcfd:ff:fe00:302.547: dhcp6 relay-reply
```

Der Relay-Agent hat also die Nachrichten, die er vom Client erhalten hat, in RELAY-FORW-Pakete verpackt und dann an die Multicastadresse ff05::1:3 versendet. Der Server wiederum hat seine Pakete an den Client in RELAY-REPL-Nachrichten verpackt und per Unicast an den Relay-Agenten versendet.

6.4.3.5 Präfix-Delegations-Beispiel

Das folgende kleine Beispiel verwendet das gleiche Szenario wie das vorangehene Relay-Agenten-Beispiel, allerdings wird auf dem Rechner „Relay“ nun ein Client gestartet und die Konfigurationsdateien aus dem Anhang 8.2.5 geladen.

Dem Client (Router) wird daraufhin vom Server ein Präfix zugeordnet und dieser teilt dies auch auf die einzelnen Schnittstellen auf, allerdings versendet er keine Router-Ankündigungsnachrichten, wie der

User's Guide zu Dibbler glauben macht, noch ist dokumentiert, wo der die Konfigurationsparameter ablegt, damit etwa Radvd darauf zugreifen könnte.

Deshalb muss die tatsächliche Demonstration der Präfixdelegation leider an dieser Stelle offen bleiben.

7 Zusammenfassung

IPv6 bietet mit der zustandslosen Autokonfiguration eine einfache automatische Konfigurationsmöglichkeit für Hosts an, die aber leider aufgrund von Designschwächen wohl selten zum Einsatz kommen wird. Denn es ist nicht möglich mit diesem Verfahren so notwendige Parameter, wie etwa die Adresse eines DNS-Servers zu konfigurieren. Es bleibt sicherlich zu hoffen, dass einer der Vorschläge aus Kapitel 5 aufgegriffen wird, damit die zustandslose Autokonfiguration doch Verwendung finden wird.

DHCPv6 hingegen bietet sehr viele Möglichkeiten, der automatischen Konfiguration. . Die Neuerungen gegenüber dem alten DHCP für IPv4 (Neues Nachrichtenformat, das die Altlasten des Bootstrap-Protokolls nun endgültig abstreifen konnte, sowie IPv6 spezifische Neuerungen wie die Präfixdelegation) sind sinnvoll. Es benötigt dafür aber einen höheren Konfigurationsaufwand.

Im Rahmen dieser Arbeit blieben Sicherheitsaspekte von IPv6 und die darauf Aufbauenden Sicherheitsmöglichkeiten von DHCPv6 unbeachtet.

8 Anhang

8.1 Szenarien

8.1.1 Szenario 1

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE vnuml SYSTEM "/usr/share/xml/vnuml/vnuml.dtd">

<vnuml>
<global>
      <version>1.8</version>
      <simulation_name>test</simulation_name>
      <ssh_version>2</ssh_version>
      <ssh_key>~/.ssh/identity.pub</ssh_key>
      <automac/>
      <netconfig promisc="on"/>
      <vm_mgmt type="private" network="10.250.0.0" mask="24">
            <host_mapping />
      </vm_mgmt>
      <vm_defaults exec_mode="net">
            <filesystem type="cow">/winhost/beta1/debian.ext3</filesystem>
            <kernel>/winhost/beta1/linux-2.6.20.1-uml0</kernel>
            <console id="0">xterm</console>
      </vm_defaults>
</global>

<!-- Netze - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -->
<net name="net0" mode="virtual_bridge" />

<!-- Hosts - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -->
<vm name="A">
      <if id="1" net="net0">
            <mac>fe:fd:a:0:0:1</mac>
      </if>
</vm>

<vm name="B">
      <if id="1" net="net0">
            <mac>fe:fd:b:0:0:1</mac>
      </if>
</vm>

<vm name="C">
      <if id="1" net="net0">
            <mac>fe:fd:c:0:0:2</mac>
```

```
      </if>
```

```
</vm>
```

```
<!-- Router  - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -  -->
<vm name="R1">
      <if id="1" net="net0">
            <mac>fe:fd:d:0:0:3</mac>
            <ipv6>3ffe::3/64</ipv6>
      </if>
      <forwarding type="ip" />
</vm>

</vnuml>
```

8.1.2 Szenario 2

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE vnuml SYSTEM "/usr/share/xml/vnuml/vnuml.dtd">
<vnuml>
<global>
      <version>1.8</version>
      <simulation_name>test</simulation_name>
      <ssh_version>2</ssh_version>
      <ssh_key>~/.ssh/identity.pub</ssh_key>
      <automac/>
      <vm_mgmt type="private" network="10.250.0.0" mask="24">
            <host_mapping />
      </vm_mgmt>
      <vm_defaults exec_mode="mconsole">
            <filesystem type="cow">/winhost/beta1/debian.ext3</filesystem>
            <kernel>/winhost/beta1/linux-2.6.20.1-uml0</kernel>
            <console id="0">xterm</console>
      </vm_defaults>
</global>

<!-- Hosts - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -  -->
<vm name="HostA">
      <filesystem type="direct">/winhost/beta1/debian.ext3</filesystem>
      <filetree seq="start" root="/install/">/winhost/patchday</filetree>
      <exec seq="stop" type="verbatim"> rm -f -r  /install/*</exec>
 </vm>

</vnuml>
```

8.1.3 Einfaches DHCP-Beispiel

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE vnuml SYSTEM "/usr/share/xml/vnuml/vnuml.dtd">
```

```
<vnuml>

  <global>
    <version>1.8</version>
    <simulation_name>test</simulation_name>
       <ssh_version>2</ssh_version>
    <ssh_key>~/.ssh/id_rsa</ssh_key>
    <automac/>
    <vm_mgmt type="private" network="10.250.0.0" mask="24">
       <host_mapping />
    </vm_mgmt>
    <vm_defaults exec_mode="net">
       <filesystem type="cow">/beta1/debian.ext3</filesystem>
       <kernel>/beta1/linux-2.6.20.1-uml0</kernel>
       <console id="0">tty</console>
    </vm_defaults>
  </global>

<!-- Netze - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -  -->
<net name="net0" mode="virtual_bridge" />


<!-- Hosts - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -  -->
<vm name="Client">
<if id="1" net="net0">
</if>
<filetree seq="start"
root="/etc/dibbler">/winhost/dhcp6test/client_1</filetree>
<exec seq="daemon" type="verbatim">dibbler-client start</exec>
</vm>

<vm name="Server">
<if id="1" net="net0"></if>
<filetree seq="start"
root="/etc/dibbler">/winhost/dhcp6test/server_1</filetree>
<exec seq="daemon" type="verbatim">dibbler-server start</exec>
</vm>


</vnuml>
```

8.1.4 Relay-Agent-Beispiel

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE vnuml SYSTEM "/usr/share/xml/vnuml/vnuml.dtd">

<vnuml>
  <global>
    <version>1.8</version>
```

```
    <simulation_name>test</simulation_name>
       <ssh_version>2</ssh_version>
    <ssh_key>~/.ssh/id_rsa</ssh_key>

    <automac/>
    <vm_mgmt type="private" network="10.250.0.0" mask="24">
       <host_mapping />
    </vm_mgmt>
    <vm_defaults exec_mode="net">
       <filesystem type="cow">/beta1/debian.ext3</filesystem>
       <kernel>/beta1/linux-2.6.20.1-uml0</kernel>
       <console id="0">tty</console>
    </vm_defaults>
  </global>

<!-- Netze - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -  -->
<net name="net1" mode="virtual_bridge" />
<net name="net2" mode="virtual_bridge" />


<!-- Hosts - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -  -->
<vm name="Client1">
      <if id="1" net="net1"></if>
      <filetree seq="start"
root="/etc/dibbler">/winhost/dhcp6test/client_4</filetree>
      <exec seq="daemon" type="verbatim">dibbler-client start</exec>
</vm>

<vm name="Client2">
      <if id="1" net="net2"></if>
      <filetree seq="start"
root="/etc/dibbler">/winhost/dhcp6test/client_4</filetree>
      <exec seq="daemon" type="verbatim">dibbler-client start</exec>
</vm>

<vm name="Relay">
      <if id="1" net="net1"></if>
      <if id="2" net="net2"></if>
      <forwarding type="ip" />
      <filetree seq="start"
root="/etc/dibbler">/winhost/dhcp6test/relay_4</filetree>
      <exec seq="daemon" type="verbatim">dibbler-relay start</exec>
</vm>

<vm name="Server">
      <if id="1" net="net2"></if>
      <filetree seq="start"
root="/etc/dibbler">/winhost/dhcp6test/server_4</filetree>
      <exec seq="daemon" type="verbatim">dibbler-server start</exec>
</vm>


</vnuml>
```

8.2 Dibbler-Konfigurationsdateien

8.2.1 4-Wege-Kommunikations-Beispiel

8.2.1.1 Client-Konfiguration

```
# client.conf
log-mode short
log-level 7
iface eth1
{
      ia
      option dns-server
}
```

8.2.1.2 Server-Konfiguration

```
# server.conf
iface eth1
{
      t1 30
      t2 60
      prefered-lifetime 90
      valid-lifetime 120
      class
      {
      pool 4711::1-4711::200
      }
      option dns-server 2000::100,2000::101
}
```

8.2.2 2-Wege-Kommunikations-Beispiel

8.2.2.1 Client-Konfiguration

```
# client.conf
log-mode short
log-level 7
iface eth1
{
      rapid-commit YES
      ia
      option dns-server
}
```

8.2.2.2 Server-Konfiguration

```
# server.conf
```

```
iface eth1
{
      t1 30
      t2 60
      prefered-lifetime 90
      valid-lifetime 120
      rapid-commit YES
      class
      {
      pool 4711::1-4711::200
      }
      option dns-server 2000::100,2000::101
}
```

8.2.3 Zustandsloses DHCP-Beispiel

8.2.3.1 Client-Konfiguration

```
# client.conf
stateless
log-mode short
log-level 7
iface eth1
{
      option dns-server
}
```

8.2.3.2 Server-Konfiguration

```
# server.conf
stateless
iface eth1
{
      option dns-server 2000::100,2000::101
}
```

8.2.4 Relay-Agent-Beispiel

8.2.4.1 Client-Konfiguration

```
# client.conf
log-mode short
log-level 7

iface eth1
{
```

```
      ia
      option dns-server

}
```

8.2.4.2 Relay-Konfiguration

```
# relay.conf
log-level 8
log-mode short

iface eth2
{
      server multicast yes
}

iface eth1
{
      client multicast yes
      interface-id 5020
}
```

8.2.4.3 Server-Konfiguration

```
# server.conf
t1 30
t2 60
prefered-lifetime 90
valid-lifetime 120
option dns-server 2000::100,2000::101

iface relay1
{
      interface-id 5020
      relay eth1
      class
      {
            pool 4000::200-4000::400
      }
}

iface eth1
{
      class
      {
            pool 4711::1-4711::200
      }
}
```

8.2.5 Präfixdelegations-Beispiel

8.2.5.1 Client-Konfiguration

```
# client.conf
log-mode short
log-level 7

iface eth2
{
      pd
      ia
      option dns-server
}
```

8.2.5.2 Server-Konfiguration

```
# server.conf
t1 30
t2 60
prefered-lifetime 90
valid-lifetime 120
option dns-server 2000::100,2000::101

iface eth1
{
      pd-class
      {
            pd-pool 4711:0:0:F000::/49
            pd-length 56
      }

      class
      {
            pool 4711::/64
      }

}
```

Literaturverzeichnis

[POTAROO] IPv4 Address Report.. 2008. http://www.potaroo.net/tools/ipv4/, zuletzt abgerufen am 15.01.2008

[HEISE] Zivadinovic, D. : Das Mega-Netz. 2007. http://www.heise.de/netze/artikel/87737, zuletzt abgerufen am 15.01.2008

[DIBBLER] Mrugalski, T. : Dibbler a portable DHCPv6 User's Guide. 2007. http://klub.com.pl/dhcpv6/doc/dibbler-user.pdf, zuletzt abgerufen am 15.01.2008

[IPV6VNUML] Schwarz, C. : IPv6 unter VNUML (Seminararbeit). 2007. für Angehörige der Universität Koblenz-Landau abrufbar unter https://www.uni-koblenz.de/~dickel/sem-ws0607/schwarz.pdf

RFCs:

Information Sciences Institute : RFC 760. Internet Protocol. 1980.

Postel, J. : RFC 768. User Datagram Protocol. 1980.

Information Sciences Institute : RFC 791. Internet Protocol. 1981.

Postel, J. : RFC 792. Internet Control Message Protocol. 1981.

Information Sciences Institute : RFC 793. Transmission Control Protocol. 1981

Callon, R. : RFC 1347. TCP and UDP with Bigger Addresses (TUBA), A Simple Proposal for Internet Addressing and Routing. 1992.

Ullmann, R. : RFC 1475. TP/IX: The Next Internet. 1993.

Fuller, V. et al. : RFC 1519. Classless Inter-Domain Routing (CIDR): an Address Assignment and Aggregation Strategy. 1993.

Droms, R. : RFC 1541. Dynamic Host Configuration Protocol. 1993.

Bradner, S. ; Mankin, A. : RFC 1550. IP: Next Generation (IPng) White Paper Solicitation. 1993.

Francis, P. : RFC 1621. Pip Near-term Architecture. 1994.

Egevang, K. ; Francis, P. : RFC 1631. The IP Network Address Translator (NAT). 1994.

Hinden, R. : RFC 1710. Simple Internet Protocol Plus White Paper. 1994.

ST2 Working Group : RFC 1819. Internet Stream Protocol Version 2 (ST2) Protocol Specification - Version ST2+. 1995.

Deering, S. ; Hinden, R. : RFC 1883. Internet Protocol, Version 6 (IPv6) Specification. 1995.

Hinden, R. ; Deering, S. : RFC 2373. IP Version 6 Addressing Architecture. 1998.

Deering, S. ; Hinden, R. : RFC 2460. Internet Protocol, Version 6 (IPv6) Specification. 1998.

Narten, T. ; Nordmark, E. ; Simpson, W. : RFC 2461. Neighbor Discovery for IP Version 6 (IPv6). 1998.

Deering, S. ; Fenner, W. ; Haberman, B. : RFC 2710. Multicast Listener Discovery (MLD) for IPv6. 1999.

Srisuresh, P. ; Egevang, K. : RFC 3022. Traditional IP Network Address Translator (Traditional NAT). 2001.

Narten, T. ; Draves, R. : RFC 3041. Privacy Extensions for Stateless Address Autoconfiguration in IPv6. 2001.

Droms, R. et al. : RFC 3315. Dynamic Host Configuration Protocol for IPv6 (DHCPv6). 2003.

Hinden, R. ; Deering, S. ; Nordmark, E.: RFC 3587. IPv6 Global Unicast Address Format. 2003.

Troan, O; Droms, R : RFC 3633. IPv6 Prefix Options for Dynamic Host Configuration Protocol (DHCP) version 6. 2003

Cheshire, S. ; Aboba, B. ; Guttman, E. : RFC 3927. Dynamic Configuration of IPv4 Link-Local Addresses. 2005.

Hinden, R. ; Haberman, B. : RFC 4193. Unique Local IPv6 Unicast Addresses. 2005.